68' MICRO JOURNAL

Australia A $4.75 | New Zealand NZ $ 6.50
Singapore S $9.45 | Hong Kong H $23.50
Malaysia M $9.45 | Sweden 30:-SEK

$2.95 USA

OS-9 Atari Amiga Mac S-50

6800 6809 68008 68000 68010 68020 68030

The Magazine for Motorola CPU Devices *For Over a Decade!*

This Issue:

Also: FORTH, Add Graphics to your SBC

OS-9 SK*DOS Atari Amiga FLEX Macintosh

A User Contributed Journal

And Lots More!

VOLUME X ISSUE XI • Devoted to the 68XXX User • November 1988

The Grandfather of "DeskTop Publishing™"

SERVING THE 68XXX USER WORLDW

000422 A/E MJ
MR. MICKEY FERGUSON
P. O. BOX 87
KINGSTON SPRINGS TN 37082

0 74470 12810 11

PHOTO CREDIT: NASA

WHO DO YOU CALL WHEN YOUR DEBUGGER WON'T DEBUG?

The problem with most real-time operating systems is simple, they're not an integrated solution. You end up dealing with a multitude of suppliers for languages, compilers, debuggers and other important development tools. And when something does go wrong, it can be a frustrating experience trying to straighten out the mess.

Why Not Try the Microware One-Stop Total Solution?

Microware's OS-9 Real-Time Operating System is a total integrated software system, not just a kernel. We offer an extensive set of development tools, languages, I/O and Kernel options. ***And this total integrated solution is entirely designed, built and supported by the same expert Microware team.***

Modularity Lets YOU Choose Just What You Need.

The modular design of OS-9 allows our Operating System to adapt as your requirements change. OS-9 can support a complete spectrum of applications — from embedded ROM-based code in board-level products all the way up to large-scale systems.

The OS-9 Success Kit

A Total Integrated Solution for Your Next Project

Development Tools:
- C Source Level Debugger
- Symbolic Debugger
- System State Debugger
- uMACS Text Editor
- Electronic Mail
- Communications
- Super Shell

Kernel Options:
- MMU (Security Protection) Support
- Math Coprocessor Support

Languages:
- C*
- Basic
- Pascal
- Fortran
- Ada**
- Assembler*

I/O Options:
- SCSI, SASI & SMD Disks
- 3-, 5-, 8-inch Diskettes
- Magnetic Tape
- Ethernet - TCP/IP
- Arcnet - OS-9/Net

*Resident or UNIX versions available
**VAX hosted

Support is Part of the Package.

Microware is proudly setting the industry's standard for customer support. You'll find professional and comprehensive technical documentation and a Customer Hotline staffed by courteous and authoritative software engineers.

So stop messing with simple kernels and independent suppliers. Call Microware today and find out more about the "One-Stop Integrated Solution" with OS-9!

microware® OS-9

Microware Systems Corporation
1900 N.W. 114th Street
Des Moines, Iowa 50322
Phone: 515/224-1929

Western Regional Office
4401 Great America Parkway
Santa Clara, California 95054
Phone: 408/980-0201

Microware Japan Ltd.
41-19 Honcho 4-Chome
Funabashi City
Chiba 273, Japan
Phone: 0474 (22) 1747

Mustang-020 Mustang-08 Benchmarks

	32 bit Integer	Register Long
IBM AT 7300 Xenix Sys 3	7	3
AT&T 7300 UNIX PC 68010	7	3
DEC VAX 11/780 UNIX Berkley 4.2	1.1	0.2
DEC VAX 11/750 " "	1.1	0.2
68008 OS-9 68K 8 Mhz	2.5	1.0
68000 OS-9 68K 10 Mhz	6.5	4.0
MUSTANG-08 68000 OS-9 68K 10 Mhz	*9.8*	*6.3*
MUSTANG-020 68020 OS-9 68K 16 Mhz	*2.2*	*0.88*
MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz	*1.8*	*1.22*

```
Main()
        {
            register long i;
            for (i=0; i < 999999; ++i);
        }
```

Estimated MIPS - MUSTANG-020 — 4.5 MIPS,

Burst to 8 - 10 MIPS: Motorola Specs

Mustang Specifications

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
32-bit wide data and address buses, non-multiplexed
on chip instruction cache
object code compatible with all 68XXX family processors
enhanced instruction set - math co-processor interface
68881 math hi-speed floating point co-processor (optional)
direct extension of full 68020 instruction set
full support IEEE P754, draft 10.0
transcendental and other scientific math functions
2 Megabyte of SIP RAM (512 x 32 bit organization)
up to 256K bytes of EPROM (64 x 32 bits)
4 Asynchronous serial I/O ports standard
optional to 20 serial ports
standard RS-232 interface
optional network interface
buffered 8 bit parallel port (1/2 MC68230)
Centronics type pinout
expansion connector for I/O devices
16 bit data path
256 byte address space
2 interrupt inputs
clock and control signals
Motorola I/O Channel Modules
time of day clock/calendar w/battery backup
controller for 2, 5 1/4" floppy disk drives
single or double side, single or double density
35 to 80 track selectable (48-96 TPI)
SASI interface
programmable periodic interrupt generator
interrupt rate from micro-seconds to seconds
highly accurate time base (5 PPM)
5 bit sense switch, readable by the CPU
Hardware single-step capability

Mustang Hi-Speed Systems
Only from Data-Comp Div.
68008-68030

Don't be mislead!
ONLY Data-Comp
delivers the Super
MUSTANG-020

OS-9

OS-9 Professional Ver *Includes C Compiler	$850.00
Basic09	300.00
C Compiler	500.00
68000 Disassembler (w/source add. $100.00)	100.00
Fortran 77	750.00
Microware Pascal	500.00
Oregon Soft Pascal	900.00
Stylo-Graph	495.00
Stylo-Spell	195.00
Stylo-Merge	175.00
Stylo-Graph-Spell-Merge	695.00
PAT w/C source	229.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Sculptor+ (see below)	995.00
COM	125.00

UniFLEX

UniFLEX (68020 ver)	$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/PreCompiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Sculptor+ (see below)	995.00

Standard MUSTANG-020™ shipped 12.5 Mhz

Add for 16.6 Mhz 68020	375.00
Add for 16.6 Mhz 68881	375.00
Add for 20 Mhz 68020/RAM	750.00
16 Port exp. RS-232	335.00
Requires 1 or 2 Adaptor Cards below RS232 Adapter	165.00

Each card supports 4 additional ser. ports (total of 36 serial ports supported)

60 line Parallel I/O card Uses 3 68230 Interface/Timer chips, 6 groups of 8 lines each, seperate buffer direction control for each group.	398.00
Prototype Board areas for both dip and PGA devices & a pre-wired memory area up to 512K DRAM.	75.00
SBC-AN Interface between the system and ARCNET modified token-passing LAN, fiber optics optional - call.	475.00
LAN software drivers	120.00
Expansion for Motorola I/O Channel Modules	$195.00

Special for complete MUSTANG-020™ system buyers - Sculptor+ $695.00, SAVE $300.00

Software Discounts

All MUSTANG-020™ systems and board buyers are entitled to discounts on all listed software: 10-70% depending on item. Call or write for quotes. Discounts apply after the sale as well.

The **PRO**!

Only the "PRO" Version of OS-9 Supported!

This is *HEAVY DUTY* Country!

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission, Government Agencies as well as Universities, Business, Labs, and other Critical Applications Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level V compatability and low cost is a must.

Price List:

Mustang-020 SBC		$2490.00
Cabinet w/switching PS		$299.95
5"-80 track floppy DS/DD		$269.95
Floppy Cable		$39.95
OS-9 68K Professional Version		$850.00
C Compiler ($500 Value)		N/C
Winchester Cable		$39.95
Winchester Drive 25 Mbyte		$895.00
Hard Disk Controller		$395.00
Shipping USA UPS		$20.00
UniFLEX	Less	$100.00
MC68881 f/p math processor	Add	$275.00
16.67 Mhz MC68020		$375.00
16.67 Mhz MC68881		$375.00
20 Mhz MC68020 Sys		$750.00
Note all 68881 chips work with 20 Mhz Sys		
	Total:	$5299.80

For a limited time we will offer a $400 trade-in on your old 68XXX SBC. Must be working properly and complete with all software, cables and documentation. Call for more information

SAVE UP TO $1500

Christmas Special Complete Systems

25 Mbyte HD ~~$4299.80~~ $3799.80

85 Mbyte HD ~~$5748.80~~ $4948.80

Note: Only Professional OS-9 Now Available (68020 Version)
Includes ($500) C Compiler - 68020 & 68881 Supported -
For UPGRADES Write or Call for Professional OS-9 Upgrade Kit

Data-Comp Division
A Decade of Quality Service
Systems World-Wide
Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6680 Hixson, Tn 37343

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Cheryl Hodge

Contributing & Associate Editors

Ron Anderson
Ron Voigts
Doug Lurie
Ed Law
Dr. E.M. "Bud" Pass
Art Weller
Dr. Theo Elbert
& Hundreds More of Us

Contents

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.
"Over a Decade of Service"
"World Wide"

68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343
Phone (615) 842-4600 Telex 510 600-6630

68 Micro Journal is the *original* "DeskTop Publishing" product and has continously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

---

68 Micro Journal (ISSN 0194-5025) is published 12 times a year by Computer Publishing Inc. Second Class Postage paid at Hixson, TN. and additional entries. POSTMASTER: send address changes to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year $24.50 USA, Canada & Mexico $34.00 a year. Others add $12.00 a year surface, $48.00 a year Airmail, USA funds. 2 years $42.50, 3 years $64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK*DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

Please - do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.

Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of $15.50 for first 15 words. Add $.60 per word thereafter. No classifieds accepted by telephone.

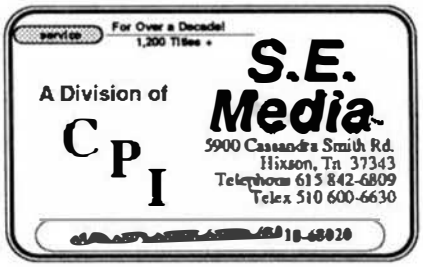
PAT - JUST

PAT

With 'C' Source

$229.00

All OS-9 68XXX Systems



PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved **PIE**, this is for **YOU! All PIE features & much more!** Too many features to list. And if you don't like ours, change or add your own. **C source included.** Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K $229.00

COMBO

PAT/JUST

Special $249.00

JUST

JUST from S. E. MEDIA - - Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The ONLY stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source $79.95

C User Notes

The C Programmers Reference Source. Always Right On Target!

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

This chapter discusses the reasons why the C language uses zero-based subscripts, rather than one-based subscripts, as used by several other popular languages. It also presents a simple file management program which illustrates the use of the C language for storage and retrieval of personal information.

SUBSCRIPTS IN C

Many popular languages, such as Fortran, Cobol, and some versions of Basic, base their subscripts at one, rather than basing them at zero. Several other languages either allow the specification of an explicit lower subscript bound or provide the option for setting the base to zero or one, but not to any other value. The original justification for this convention is lost, but was probably based on mathematical sequence notation, which typically starts with one.

Historically, there have been certain other means of computing subscript values, usually based closely upon particular hardware, which resulted in reversed or non-sequential mapping of subscripts to addresses. Luckily, no popular current systems allocate one-dimensional subscripts in other than an ascending linear fashion.

From a compiler-writer's viewpoint, the most logical subscript base value is zero, since any other value requires either a subtraction to convert the subscript to an offset from a base address or requires the computation of an offset base address such that the first element of the subscripted item will actually reside at the desired address.

In the C language, subscripting is considered another notation for indexing from a pointer. Thus, the following program fragments:

```
char *p;
char q[10];

*(p + 2) = 'a';
*(q + 3) = 'b';
```

and

```
char *p;
char q[10];

p[2] = 'a';
q[3] = 'b';
```

would produce equivalent results.

Since adding zero to a pointer is equivalent to not offsetting the pointer with a value, subscripts in the C language are automatically zero-based. Although this might seem strange to a Cobol or Fortran programmer who has always used one-based subscripts, it is a natural outgrowth of the definition of the C language subscript notation.

It is not entirely possible for a programmer to ignore the distinction between zero-based subscripting and one-based subscripting, which might be the tendency for one familiar with one-based

subscripting who must program in a language with zero-based subscripts.

The existence of element zero usually may be safely ignored, except in certain cases. The wasted space will be important only in cases in which memory space is tight, and then only in multi-dimensional cases. It may never be ignored in cases in which functions coded in languages with different subscript bases must pass information internally.

An area which will always be different concerns the declared size of the subscript range. In Basic, Cobol, and Fortran, the declared size corresponds to the upper bound of the subscript. The lower bound is normally one, although in some cases, it is allowed to be zero. In the C language, the size is the actual number of elements, so that it will be one more than the upper bound of legal values of the subscript. Thus, the following Basic program fragment:

```
DIM Q(10)
```

would properly be translated into the following C program fragment:

```
char q[11];
```

Assuming that the element with subscript zero is not being used.

INTRODUCTION TO RF - *Rotary File Program*

The program presented as an example below was developed and placed into the public domain by Larry Lippman, of Recognition Research Corporation of Clarence, New York.

Lippman calls this program rf. It is a simple "rotary file" file management system which stores names, addresses, telephone numbers, comments, etc. for rapid retrieval in a formatted page fashion.

The rf program searches its input file using a search string keyed to a person's name or person's organization. Multiple matches are paginated using a single key command.

In its present form, it will search for a record by individual name or by organization name. Any size search string may be used as the key. Command-line parsing is intelligent enough such that quotes are unnecessary if the search string contains embedded whitespace. The output is displayed in a page-oriented fashion, with paging control used when multiple matches are encountered.

The input file is easily configured using any editor capable of creating files devoid of proprietary formats or character sequences. The input file format is also designed so that it may be accessed by awk or sed for mailing list or other special applications. By setting an option, the user may access a system-wide file or a private file in their home directory.

While the program could have been written using awk and tput, it is substantially faster and more portable because it is written in C. Typical search and display (9,600 baud) time on an AT&T 3B2 for one record from a 30 kilobyte file is less than 2 seconds. Since this includes curses overhead, it's reasonably fast. The entire file is always searched to detect multiple matches.

The program is reasonably well-commented, is intended to be easily modifiable, and is reasonably well-protected against users doing "dumb" things. The program runs on three different unix Sys V versions, and should probably run under bsd unix since it contains its own string search function and does not use getopt. The curses use is not particularly exotic, and could be replaced without too much difficulty on systems without curses or with versions which do not properly handle common terminals.

The actual fields and field lengths were chosen to provide a program that fit the needs of Lippman's organization - an industrial R&D laboratory which communicates a great deal with various other organizations.

The program is written so that the field definitions and their lengths can be easily modified for a particular organization. For example, an organization might often send telex and facsimile machine

messages, but other organizations never send such messages - so these fields could be removed, allowing character expansion of the telephone number and uucp address fields.

Some people may not like the choices of video attributes for the page display; obviously, this is easy to change.

DESCRIPTION OF RF - *Rotary File Program*

The rf program has a command line of the following format:

rf [-f] [-l] [-o] [search string]

The -f flag selects a private file installed as .rf_data in the user's home directory. Invoking rf without this flag selects the file /usr/local/lib/rf_data available to all users.

The -l flag lists only the name or organization field matches without displaying the rest of the record, and is used for rapid scanning of the file where multiple matches may occur.

The -o flag searches for a match in the organization field, rather than the name field.

The search string may be composed of 1 to 72 characters. It may contain one or more instances of whitespace without having the string enclosed in quotes. There is no upper to lower or lower to upper case conversion; the case presented is matched as-is, and may be mixed. Punctuation and whitespace embedded in the string is also matched; each occurrence of whitespace must be limited to a length of one space or it may not be matched properly.

DATABASE FILE FORMAT

Each record consists of a minimum of two fields, with all fields containing a two-character identifier in the form of a letter followed by a colon. The data portion of the field may contain whitespace or any punctuation to a maximum character length as described below. Each record must begin with a name field; if there is an organization field, it must immediately follow the name field. If the record pertains to an organization only having no person's name entry, the name field identifier is still necessary, with the rest of the field blank. All other fields are optional and may be included in any order. Records are separated by one blank line.

Each field is limited to one entry per record, except that telephone, telex, fax and uucp may each have a maximum of two entries, for line-order display as presented within the record; a maximum of four comment entries is also permitted within the same record.

```
N:name                  30 chars max.
O:organization          72 chars max.
T:title                 24 chars max.
D:department            24 chars max.
A:address               72 chars max.
P:telephone             15 chars max.
F:fax                   15 chars max.
X:telex                 24 chars max.
U:uucp                  24 chars max.
H:home_telephone        15 chars max.
R:home_address          72 chars max.
```

Fields which exceed the above maximum number of characters result in no error, but are silently truncated at the maximum permissible length when displayed. Fields which contain incorrect header characters are ignored. For the sake of uniformity, comments in the file that are not intended for display should be prefaced by the header #:.

EXAMPLE FILE RECORD

```
N:Public, John Q.
O:Any Industry, Inc.
T:Systems Programmer
D:Widget R&D
A:123 Any Road, Anytown, NY 12345
P:716/123-4567
P:Ext 234
F:716/123-4599
X:12-3456 ANYINDNY
U:jqp@any.UUCP
H:716/123-9876
R:456 Anybrook Lane, Anysuburb, NY 12354
C:Writes CAD software for widgets
C:Has extensive experience with XYZNIX
```

The following files are used by the rf program:

```
$HOME/.rf_data          user private file
/usr/local/lib/rf_data  system-wide file
```

C PROBLEM, EXAMPLE C PROGRAM

Following is this month's example C program: it implements the "rotary file" system described earlier.

```
/*
 *        A "rotary file"-like file for
 *        names, addresses, telephone
 *        numbers and related information.
 *
 *        Copyright (c) 1985
 *        by Lawrence Lippman, larry@kitty.UUCP
 *        Recognition Research Corp., Clarence, NY
 *        Telephone: 716/688-1231
 *        uucp: {allegra|ames|boulder|decvax|
 *            rutgers|watmath}!sunybcs!kitty!larry
 *              {hplabs|ihnp4|mtune|utzoo|uunet}!
 *            sunybcs!kitty!larry
 *
 */

#include <stdio.h>
#include <curses.h>
#include <signal.h>
#include <string.h>

#ifndef SYSDATA
/* system file */
#define SYSDATA "/usr/local/rf_data"
#endif

/*
 * Global variables
 */

FILE *fl;
FILE *fopen();
char Key[73];
char adr[73];
char buf[80];
char comments[4][73];
char dept[25];
char fax[2][16];
char homeadr[65];
char homephone[16];
char name[31];
char org[73];
char phone[2][16];
char telex[2][25];
char title[25];
char uucp[2][25];
int File;
int List;
int Org;
int comlines;
int hits;
int more();
int phlines;
int strsearch();
int terminate();

main(argc, argv)
int argc;
char **argv;
{
    char *getenv();
    char filename[65];
    int keywords;

    /*
     * Get options and search string
     */

    (void)strcpy(Key, "");
    keywords = 0;

    while (argc-- > 1)
    {
        if (*argv[1] == '-')
            switch (argv[1][1] | 0x20)
            {
            case 'l':
                List = 1;
                break;
            case 'f':
                File = 1;
                break;
            case 'o':
                Org = 1;
                break;
            default:
                usage();
            }
        else
        {
            if (keywords > 0)
```

```
            (void)strcat(Key, " ");
        (void)strcat(Key, argv[1]);
        keywords++;
    }
    argv++;
}

if (!strlen(Key))
    usage();

/*
 * Select and open file
 */

if (File)
    (void)sprintf(filename,
        "%s/.rf_data", getenv("HOME"));
else
    (void)strcpy(filename, SYSDATA);

if (!(fl = fopen(filename, "r")))
{
    (void)fprintf(stderr,
        "Cannot open data file %s\n",
        filename);
    exit(-1);
}

/*
 * Catch signals
 */

(void)signal(SIGINT, terminate);
(void)signal(SIGQUIT, terminate);

/*
 * Initialize curses
 */

initscr();
if (List)
{
    idlok(stdscr, 1);
    setscrreg(0, 19);
    scrollok(stdscr, 1);
}

/*
 * Read file, search for records
 */

hits = 0;
while (fgets(buf, 80, fl))
{
    if (buf[0] == 'N' && buf[1] == ':')
    {
        if (strlen(buf) == 2)
            (void)strcpy(name, "");
        else
        {
            (void)strxcpy(name, buf, 2, 30);
            if (!Org)
            {
                if (strsearch(name, Key))
                {
                    if (hits > 0 && !List)
                        (void)more();
                    (void)strcpy(org, "");
                    (void)clrrecord();
                    (void)rdrecord();
                    (void)display();
                    hits++;
                }
            }
        }
    }
    if (buf[0] == 'O' && buf[1] == ':')
    {
        (void)strxcpy(org, buf, 2, 72);
        if (Org)
        {
            if (strsearch(org, Key))
            {
                if (hits > 0 && !List)
                    (void)more();
                (void)clrrecord();
                (void)rdrecord();
                (void)display();
                hits++;
            }
        }
    }
}

(void)terminate();
return(0);
```

```
}
/*
 * clrrecord:    clear all display strings
 */
clrrecord()
{
    int i;

    (void)strcpy(title, "");
    (void)strcpy(dept, "");
    (void)strcpy(adr, "");
    (void)strcpy(homephone, "");
    (void)strcpy(homeadr, "");
    (void)strcpy(comments[0], "");
    for (i = 0; i <= 1; i++)
    {
        (void)strcpy(phone[i], "");
        (void)strcpy(telex[i], "");
        (void)strcpy(fax[i], "");
        (void)strcpy(uucp[i], "");
    }
}
/*
 * rdrecord:    read the rest of a record
 */
rdrecord()
{
    int fxln;
    int phln;
    int txln;
    int uuln;

    phln = fxln = txln = uuln = 0;
    phlines = comlines = 0;

    while (fgets(buf, 80, fl))
    {

        if (buf[1] != ':' || strlen(buf) <= 1)
            return;

        switch (buf[0])
        {
        case 'O':
            (void)strxcpy(org, buf, 2, 72);
            break;
        case 'T':
            (void)strxcpy(title, buf, 2, 24);
            break;
        case 'D':
            (void)strxcpy(dept, buf, 2, 24);
            break;
        case 'A':
            (void)strxcpy(adr, buf, 2, 72);
            break;
        case 'P':
            if (phln > 1)
                break;
            (void)strxcpy(phone[phln], buf, 2, 15);
            phln++;
            break;
        case 'X':
            if (txln > 1)
                break;
            (void)strxcpy(telex[txln], buf, 2, 24);
            txln++;
            break;
        case 'F':
            if (fxln > 1)
                break;
            (void)strxcpy(fax[fxln], buf, 2, 15);
            fxln++;
            break;
        case 'U':
            if (uuln > 1)
                break;
            (void)strxcpy(uucp[uuln], buf, 2, 24);
            uuln++;
            break;
        case 'H':
            (void)strxcpy(homephone, buf, 2, 15);
            break;
        case 'R':
            (void)strxcpy(homeadr, buf, 2, 64);
            break;
        case 'C':
            if (comlines > 3)
                break;
            (void)strxcpy(comments[comlines], buf, 2, 72);
            comlines++;
            break;
        }
        if (phln > 1 || txln > 1 || fxln > 1 || uuln > 1)
            phlines = 1;
    }
}
```

```
/*
 * display:    Display the results of a search
 */
display()
{
    int l;

    if (List)
    {
        if (!hits)
        {
            erase();
            refresh();
            if (Org)
                mvaddstr(0, 0, org);
            else
                mvaddstr(0, 0, name);
        }
        else
        {
            if (Org)
                printw("%s", org);
            else
                printw("%s", name);

        }
        refresh();
        return(0);
    }

    erase();
    refresh();

    attron(A_REVERSE);
    mvaddstr(0, 0, "NAME");
    mvaddstr(0, 32, "TITLE");
    mvaddstr(0, 56, "DEPT");
    mvaddstr(3, 0, "ORGANIZATION NAME & ADDRESS");
    mvaddstr(7, 0, "TELEPHONE");
    mvaddstr(7, 16, "TELECOPIER");
    mvaddstr(7, 32, "TELEX");
    mvaddstr(7, 56, "UUCP");
    mvaddstr(10 + phlines, 0, "HOME TELEPHONE");
    mvaddstr(10 + phlines, 16, "HOME ADDRESS");
    mvaddstr(13 + phlines, 0, "COMMENTS");
    attroff(A_REVERSE);

    mvaddstr(1, 0, name);
    mvaddstr(1, 32, title);
    mvaddstr(1, 56, dept);
    mvaddstr(4, 0, org);
    mvaddstr(5, 0, adr);
    for (l = 0; l <= phlines; l++)
    {
        mvaddstr(8 + l, 0, phone[l]);
        mvaddstr(8 + l, 16, fax[l]);
        mvaddstr(8 + l, 32, telex[l]);
        mvaddstr(8 + l, 56, uucp[l]);
    }
    mvaddstr(11 + phlines, 0, homephone);
    mvaddstr(11 + phlines, 16, homeadr);
    for (l = 0; l <= comlines; l++)
        mvaddstr(14 + phlines + 1, 0, comments[l]);

    refresh();
    return(0);
}
/*
 * more:    Prompt for display of entries
 */
more()
{
again:
    attron(A_REVERSE);
    mvaddstr(20, 0, "MORE HITS: CONTINUE?");
    attroff(A_REVERSE);
    mvaddstr(20, 22, "[y] [n]  ");
    refresh();
    switch (getch() | 0x20)
    {
    case 'n':
        (void)terminate();
    case 'y':
        return(0);
    }
    mvaddstr(20, 31, " ");
    goto again;
}
/*
 * terminate:    Exit gracefully
 */
terminate()
{
    (void)fclose(fl);
    move(LINES - 1, 0);
    refresh();
```

```
    endwin();
    exit(0);
}
/*
 * strsearch:    Search for any occurrence of
 *               string "t" in string "s";
 *               return 1 if a match found,
 *               otherwise return 0
 */
strsearch(s, t)
char s[80];
char t[80];
{
    char st[80];
    int i;
    int n;
    int nn;
    int slength;
    int tlength;

    slength = strlen(s);
    tlength = strlen(t);
    if (!slength || !tlength)
        return(0);

    for (i = 0; i < slength; i++)
    {
        for (n = 0, nn = i; nn < i + tlength; nn++)
        {
            if (nn > slength)
                return(0);
            st[n++] = s[nn];
        }
        st[n] = 0;
        if (!strcmp(st, t))
            return(1);
    }
    return(0);
}
/*
 * strxcpy:    Copy string "t" to string "s",
 *             with "offset" characters in
 *             string "t" skipped before the copy,
 *             and with a maximum of "maxs"
 *             characters [not including NULL]
 *             copied to string "s"
 */
strxcpy(s, t, offset, maxs)
char s[80];
char t[80];
int maxs;
int offset;
{
    int n;
    int nn;
    int tlength;

    nn = 0;
    tlength = strlen(t);

    for (n = offset; n <= tlength + 1; n++)
    {
        if (!(s[nn] = t[n]))
            return(0);
        if (++nn >= maxs)
        {
            s[nn + 1] = 0;
            return(0);
        }
    }
    return(0);
}
/*
 * usage:    Display usage error message and exit
 */
usage()
{
    (void)fprintf(stderr,
 "usage: rf [-f] [-l] [-o] searchstring\n");
    exit(-1);
}
```

EOF

Basically OS-9

Dedicated to the serious OS-9 user.
The fastest growing users group world-wide!
6809 - 68020

A Tutorial Series

By: Ron Voigts
2024 Baldwin Court
Glendale Heights, IL 60139

XMODE, TMODE and DMODE

Well for the last few columns I have been beating on devices. I have covered RBF and SCF devices. This month will rap things up with a discussion of XMODE and TMODE. The title of this month's column suggests that DMODE also exists.

XMODE is a handy program that allows the SCF device descriptors in memory to be altered or displayed. This can be used to effect any SCF device such as video terminal, printer and RS-232 ports. Its syntax is:

XMODE device_name [parameter_list]
Device_name can be any legitimate device like /TERM, /T1 and /P. The parameter_list includes:
upc.....diplay only uppercase characters.
-upc.....display uppercase and lowercase characters.
bsb.....backspace erases characters.
-bsb.....backspace over characters.
bsl.....backspace over line.
-bsl.....backspace to next line.
echo.....echo input characters.
-echo.....no echo of characters.
lf.....add line feed with carriage return.
-lf.....no line feed.
pause.....pause at end of page.
-pause....no pause.
null=n....n is the number of nulls sent after carriage return.
pag=n.....n is the page length.
bsp=h.....h is the input backspace character in hex. Usually it is $08 or ^H.
bse=h.....h is the output backspace character in hex. Usually it is $08 or ^H.
del=h.....h is the input delete line character. Usually it is $18 or ^X.
bell=h....h is the the bell ring character. Usually it is $07 or ^G.
eor=h.....h is the end of record caracter. Usually it is $0D, ^M or a carriage return.
eof=h.....h is the the end of file character. Usually it is $1B or the escape key.
type=h....h sets the parity, modem kill status and not ready delay for any external RS-232 devices. To use this the following tables can be used.
Parity None....0 Mark...A0 Space..E0 Even...60 Odd....20
Modem Kill On....10 Off....0

Not Ready Delay number of seconds in hex i.e. 10 seconds is 0A
To find the value for h select the features you want and add them together. For example, to get Even Parity, Modem Kill On and 10 seconds, use

h=60+10+0A=7A
reprint=h..h is the reprint character in hex. Usually this is $04 or ^D.
dup=h.....h is the duplicate last line character. Usually this is $01 or ^A.
psc=h.....h is the pause character. Usually this is $17 or ^W.
abort=h..h is the terminate character. Usually this is $03 or ^C.
quit=h..h is the quit character. Usually this is $05 or ^E.
baud=h..h is the baud rate, word length and stop bits.

Using the following tables can help calculate the value for h.
Baud
0....110
1....300
2....600
3...1200
4...2400
5...4800
6...9600
7..19200 or 32000 hardware dependent

Word Length
0....8 bits
20....7 bits
Stop bits
0....1 stop bit
80..2 stop bits

To find the correct value for h, select the values you desire and add them up. For example the baud rate is 4800, with 7 bit wore and 2 stop bits. To find h pick the values out of the tables and add them.

h=5+20+80=A5

Using XMODE is just as simple. Let us say we want to change /P1 using the values of TYPE and BAUD calculated before and have it print in only uppercase characters. The input would look like:

OS9:XMODE /P1 UPC TYPE=A7 BAUD=A5

Now the P1 device descriptor would be set to the values shown here. When a path was opened to /P1, these values would be in effect. The key is that the device descriptor is changed not the path.

TMODE is used to change parameters of a path. Generally it is used on the standard paths. It does not effect the descriptor, only the path. It uses the same parameters as the XMODE. Its syntax is :

TMODE [path_number] [parameter_list]
Normally you use it from the keyboard. A line like:

OS9:TMODE -UPC -PAUSE

will allow your terminal to print upper and lower case, and turn on the page pause feature.

AddingTMODE to the STARTUP file is a little trickey. The problem is that TMODE is getting its input from the disk where the start up file is stored. For TMODE this is its standard input. Try this. Add TMODE to you startup file and watch the display when STARTUP is run. It will print something like:

/D1 upc -bsb -bsl echo -lf null=35 pause pag=0 bsp=00
del=12 eor=00 eof=12 reprint=03 dup=08 psc=00 abort=80
quit=80 bse=00 bell=0B type=00 baud=00 xon=00 xoff=13

Notice how the first line is /D1. This is what TMODE read. As result most of the information here is nonsensical. To use it in a file like STARTUP, create the line:

TMODE .1

Add any other parameters to the line as you wish. This line tells TMODE to use the path #1 for it input. This will work.

So far I have touched on 2 methods to change a SCF device or it path. This month I offer a program written in C that will change the descriptor of an RBF device. It is called DMODE and appears in Listing 1.

Enter DMODE by itself and it will print a help screen explaining the possible choices. It does reqire knowledge of what value you want to change. For example if you were to go from a single sided disk to a double sided one you would enter.

DMODE /D0 SID=2

This would go to the device desciptor change the byte at $19 to 2. It also corrects the CRC of the module so you can use DMODE and later use COBBLER to create a new disk.

One more piece of news for this column. I have finished the monumental task of pulling together all the listings for all the columns. The listings include 3 years of this column from April 1985 to March 1987. It took 3 disk and they are full! I have all the original source codes on these disks. They are either compiled, packed or assembled for your convenience. I won't do a review here, but rather tell you to give Don a call. Do to the lag time of when I write the column and it appears, it may already be available. So watch for it!

LISTING 1

```
0001 /* ***********************
0002
0003    Name: dmode.c
0004    Date: 27-JUN-88
0005    Author: Ron Voigts
0006    To Compile:  ccl dmode.c
0007
0008    ***********************
0009
0010    Version 1.0  27-JUN-88  RDV
0011    Original
0012
0013    Version 1.1  22-AUG-88  RDV
0014    Added help function.
0015
0016    ***********************
0017
0018    Function:
0019    DMODE allows the device descriptor
0020    values to be changed. It does
0021    require knowledge of desired values.
0022
0023    *********************** */
0024
0025 #include <ctype.h>
0026 #include <stdio.h>
0027 #include <module.h>
0028
0029 static char *p[] = {
0030      "DTP", "DRV", "STP",
0031      "TYP", "DNS", "CYL",
0032      "SID", "VFY", "SCT",
0033      "TOS", "ILV", "SAS"
0034 };
0035
0036 /* Main program */
0037  main(argc,argv)
0038 int argc;
0039 char **argv;
0040
0041 {
0042 /* Structure for module header */
0043       struct header {
0044           unsigned sync;
0045           unsigned size;
0046           unsigned offset;
0047           char ty_at ;
0048           char la_rv;
0049           char pc;
0050           unsigned fmname;
0051           unsigned ddname;
0052           char mode;
0053           char control[3];
0054           char tabsize;
0055      };
0056
0057 /* Structure for device parameters */
0058      struct parameters {
0059           char dtp;
0060           char drv;
0061           char stp;
0062           char typ;
0063           char dns;
0064           unsigned cyl;
0065           char sid;
0066           char vfy;
0067           unsigned sct;
0068           unsigned tos;
0069           char ilv;
0070           char sas;
0071      };
0072
0073 /* Declare variables used */
0074      struct header *mod;
0075      struct parameters *it;
0076      mod_dev *modlink();
0077      mod_dev *munlink();
0078      int i;
0079
0080      /* If no arugments print help message */
0081      if ( argc==1 )
0082           help();
0083      /* If 1 argument print information about the device */
0084      else if ( argc==2 ) {
0085           /* Not a device name */
0086           if ( argv[1][0] != '/' )
0087                help();
0088           /* Get the parameters and print them */
0089           mod = modlink(argv[1]+1, 15, 1 );
0090           if ( mod == -1 )
0091                exit( errno );
```

```
0092          it=(int)mod+sizeof(*mod);
0093          printf("DRV=%2d ",(int)(it->drv));
0094          printf("STP=%2d ",(int)(it->stp));
0095          printf("TYP=%2d ",(int)(it->typ));
0096          printf("\n");
0097          printf("DNS=%2d ",(int)(it->dns));
0098          printf("CYL=%2d ",it->cyl);
0099          printf("SID=%2d ",it->sid);
0100          printf("\n");
0101          printf("VFY=%2d ",it->vfy);
0102          printf("SCT=%2d ",it->sct);
0103          printf("TOS=%2d ",it->tos);
0104          printf("\n");
0105          printf("ILV=%2d ",it->ilv);
0106          printf("SAS=%2d ",it->sas);
0107          printf("\n");
0108          munlink( argv[1]+1 );
0109     }
0110     /* Else the device descriptor is to be changed */
0111     else {
0112          mod = modlink(argv[1]+1, 15, 1 );
0113          if ( mod == -1 )
0114               exit( ermo );
0115          it=(int)mod+sizeof(*mod);
0116          /* Proces the remaining parameters */
0117          for (i=2; i<argc; i++)
0118               switch( isin( argv[i] ) ) {
0119                    case 0:
0120                         printf("Can't change device type\n");
0121                         break;
0122                    case 1:
0123                         it->drv=(char)value(argv[i]);
0124                         break;
0125                    case 2:
0126                         it->stp=(char)value(argv[i]);
0127                         break;
0128                    case 3:
0129                         it->typ=(char)value(argv[i]);
0130                         break;
0131                    case 4:
0132                         it->dns=(char)value(argv[i]);
0133                         break;
0134                    case 5:
0135                         it->cyl=value(argv[i]);
0136                         break;
0137                    case 6:
0138                         it->sid=value(argv[i]);
0139                         break;
0140                    case 7:
0141                         it->vfy=value(argv[i]);
0142                         break;
0143                    case 8:
0144                         it->sct=value(argv[i]);
0145                         break;
0146                    case 9:
0147                         it->tos=value(argv[i]);
0148                         break;
0149                    case 10:
0150                         it->ilv=value(argv[i]);
0151                         break;
0152                    case 11:
0153                         it->sas=value(argv[i]);
0154                         break;
0155                    default:
0156                         printf("No default value\n");
0157                         break;
0158               }
0159               /* Fix the module CRC in memory */
0160               fixcrc( mod, mod->size);
0161     }
0162 }
0163
0164 /* Help message */
0165 help()
0166 {
0167      printf("Usage:\n");
0168      printf("dmode [RBF_Device] [Parameter_List]\n");
0169      printf("Parameters:\n");
0170      printf("DRV = Drive number\n");
0171      printf("STP = Step rate\n");
0172      printf("TYP = Device type\n");
0173      printf("DNS = Media density\n");
0174      printf("CYL = Numer of cylinders\n");
0175      printf("SID = Number of sides\n");
0176      printf("VFY = Verify disk writes\n");
0177      printf("SCT = Sectors per track\n");
0178      printf("TOS = Sector on track 0\n");
0179      printf("ILV = Interleave factor\n");
0180      printf("SAS = Segment allocation\n");
0181      exit( 0 );
0182 }
0183
```

```
0184 /* Returns the table location of from the argument */
0185 isin( s )
0186 char *s;
0187 (
0188      int i;
0189      upper( s );
0190      for ( i=0; i<12; i++ )
0191           if ( findstr(1,s,p[i]) )
0192                return( i );
0193      return( -1 );
0194 }
0195
0196 /* Returns integer value from argument */
0197 value( s )
0198 char *s;
0199 {
0200      while ( *s!='=' ) [
0201           if ( *s=='\0' )
0202                return( 0 );
0203           s++;
0204      }
0205      return( atoi( ++s ) );
0206 }
0207
0208 /* Cconvert string to upper case */
0209 upper( t )
0210 char *t;
0211 (
0212      while( *t!='\0' ) [
0213           *t=toupper( *t );
0214           t++;
0215      }
0216 }
0217
0218 /* Fix CRC of module in memory */
0219 fixcrc( start, size)
0220 char *start;
0221 int size;
0222 {
0223      char c[3];
0224      int i;
0225      i=start+size-sizeof[c];
0226      c[0]=c[1]=c[2]=255;
0227      crc( start, size-3, c );
0228      c[0]=~c[0];
0229      c[1]=~c[1];
0230      c[2]=~c[2];
0231      start[size-3]=c[0];
0232      start[size-2]=c[1];
0233      start[size-1]=c[2];
0234 }
0235
```

Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - *what you want!*

The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2

There was no official TEST last time, so, as I intend you to try those same exercises as TEST TWELVE I'll leave the solutions till another time.

Are we all gathered round? Pencils and squared-paper at the ready? OK then, here we are at

Mile 15 - heading for Mile 16

SYMMETRIC FUNCTIONS (continued)

SHIFTING DOWN

When symmetric notations have multiple subscripts with a difference GREATER than 1 and THE NEXT NUMBER IN THE PROGRESSION WOULD BE GREATER THAN THE NUMBER OF VARIABLES OF SYMMETRY, a process known as "shifting down" can be employed to substantially reduce the number of transfers in the circuit-design.

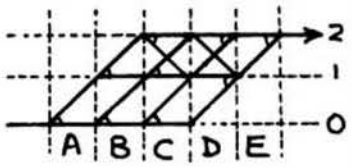


Diagram 74

During the previous mile of our journey (diagram 71) we designed a S_2^5,4 ABCDE using basic principles. Now we're going to re-design it using the shifting-down technique. We can shift-down because

(a) the subscripts have a difference greater than 1, and
(b) the next subscript in sequence is greater than the superscript.

That is, the next number in the sequence 2, 4 .... is 6, which is greater than the superscript "5".

Our first step is to draw the "fish-net" for the lowest subscript (in this case, 2-out-of-5). Then, instead of superimposing a 4-out-of-5 parallelogram, we shift-down through NO-contacts to the 1-level, as shown in Diagram 74, using the following procedure.

First, we must calculate the number of levels through which to shift down, this being ALWAYS one less

than the difference between the subscripts, ie 4 - 2 = 2 (the difference) and 2 - 1 = 1, so we'll shift-down through one level. This is done by commencing in the top-left corner of the parallelogram, and slanting downwards to the right ACROSS ONE COLUMN ONLY to level-1 (one level down). Then we move to the right along the top edge of the parallelogram to the next relay's column, similarly slanting downwards to the right, and continuing until there are no more meaningful points to which to connect. For instance, we don't add a third shift-down in the E-column in our example, as the lower end of this line would not be attached to any part of the parallelogram at the 1-level.

The circuit action is visualised thus :

Commencing at the bottom-left corner (power-input), zero relays operated would confine us to the 0-level, one relay operated would move us up to level-1, and two relays operated to level-2 and out to the output-line. But, if a third relay operated it would shift us back down to level-1 (now properly called the 1,3-level) cutting off the output in so doing. A fourth relay would send us back up to level-2 (now the 2,4-level), re-activating the output, while the fifth and final relay would cut it off again by returning us to the 1,3-level (ultimately called the 1,3,5-level!).

The diagonal-lines added for shifting-down are interpreted as a SINGLE NO-CONTACT on the relay in whose column they appear, NO MATTER HOW LONG THEY ARE!

If all this still seems a little hard to accept, you can again pick any combination of 2 or 4 relays and verify that the output will be activated, but not if ANY other combination is selected!

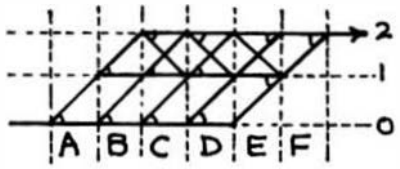


Diagram 75

Diagram 75, which attempts to employ shift-down for a $S\,^{6}_{2}$,4 ABCDEF, shows why this method cannot be used if the next step in the subscript-progression is NOT greater than the superscript. Interpreting the circuit action as before, we see that an output would be obtained if any two relays operated; we'd be back down at level-1 for 3; output at level-2 again for 4, down again to level-1 for 5 relays, and finally UP TO LEVEL-2 FOR 6 RELAYS, AND OUT AT THE OUTPUT-LINE. This final output is not called for in the specs, the circuit in effect being a $S\,^{6}_{2}$,4,6 ABCDEF network.

MULTIPLE-LEVEL SHIFTING-DOWN

A shift-down may be made through more than one level, as long as the number of levels shifted is one less than the difference in the subscript progression. For example, let's design a $S\,^{8}_{3}$, 7 network.

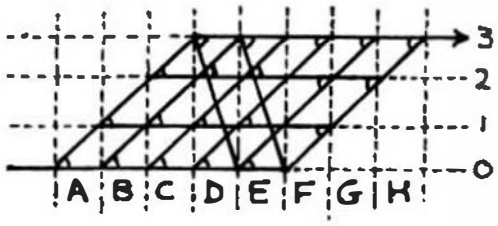


Diagram 76

As before, we design for the lowest subscript, that is a 3-out-of-8 circuit. Then, calculating one less than the difference between the subscripts (7 - 3 - 1 = 3), we shift-down through 3 levels, following the procedure described, using two NO-contacts (one on relay-D and one on E), which are already available on existing transfers, but unused, and hey presto! we've achieved the 7-out-of-8 part of the specs! If we'd followed basic principles instead, and superimposed a 7-out-of-8 parallelogram over the original 3-out-of-8, quite a large number of extra contacts would have been necessary. Try it and see!!!

Sometimes, though, it's necessary to add levels before the required shift-down can be made, as in the case of the network for $S^{8}_{2},7$ ... coming up next.

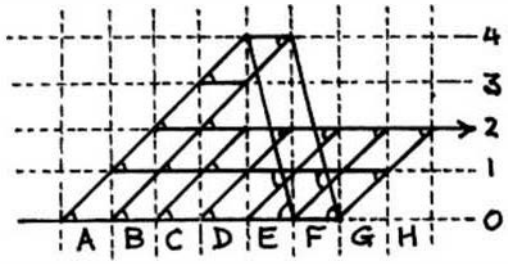


Diagram 77

Refer to Diagram 77, where we've drawn the lowest 2-out-of-8 network. Calculating the shift-down results in 7 - 2 - 1 = 4, but unfortunately we only have 2 levels through which to shift. So let's produce 4-levels by slanting up A SINGLE LINE from the lower-left corner to level-4, and from there we'll shift-down by four levels to level-0 at the right-hand side of relay-E. H'm! We see that it's possible to shift-down across relay-F too, so we'll extend our new parallelogram along level-4, and add this new NO-contact to relay-F. There's no more shift-down possible, otherwise the next would have to connect to the right-hand side of relay-G at level-0, and there's no valid connection-point there. So we merely "fill-in" the new parallelogram slanting upwards from level-2 to level-4, BUT WE DON'T FILL-IN BETWEEN THE SHIFT-DOWN LINES ... NOT EVER!!

Now for yet another technique which allows us to save relay-contacts. This relies on the identification of

EQUIVALENT POINTS

Let's design a $S^{5}_{1},3$ ABCDE. The first thing we realise is that we can't use a shift-down, as the next step in the progression is NOT greater than the superscript 5, and so we construct the network of Diagram 78a.

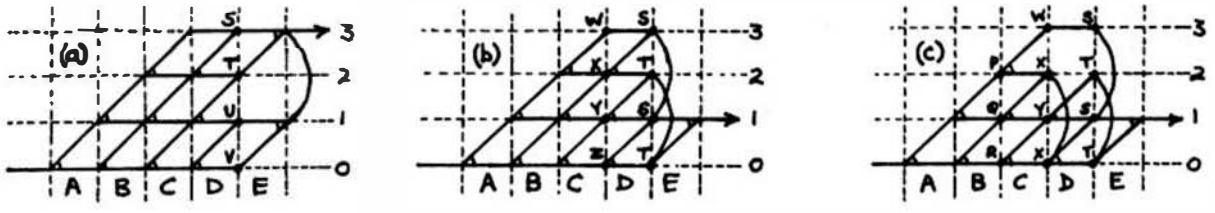


Diagram 78

Commencing at the output end (at both levels 1 and 3) and looking back through relay-E's contacts, we can see the points S, T, U and V, which we mark on the network. Then we draw up the table of Diagram 79a, with the rows labelled for the four points under consideration, and the columns headed "E = 0" and "E = 1". Now, considering point-S, we note that if relay-E is NOT operated, then S is connected (via the horizontal NC-contact of E) to the output, which we label as "Z", and if relay-E IS operated then S goes nowhere. These facts are recorded in the row labelled S. Point-T, on the other hand, goes nowhere when E = 0 (cut off by E's NO-contact) but is connected to Z when E = 1. This, too, is noted in the table, and so on for points U and V.

Having done all that, let's look at 79a! Right away we see that S and U are equivalent, as are T and V. Our new procedure allows us to join S to U and T to V, and to eliminate any contacts leading from S and T to the curved lines of the output, as shown in 78b. Remembering what I said earlier about output-lines only being allowed from the right-hand side of our graph, we are left with only the output from level-1. We could equally as well have eliminated the contacts going to the output from U and V, which would have left us with a single output-line at level-3 instead. Anyway, the net result is that we've saved one transfer on relay-E.

But we're not done yet! Point-S now exists on both levels 1 and 3, and point-T on levels 0 and 2, so let's look back from these points on the two curved lines just inserted, through relay-D's contacts to new points W, X, Y and Z (this "Z" won't be confused with the output-line as none of these points can directly reach the output any more).

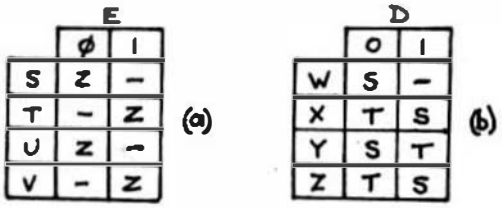


Diagram 79

We therefore draw up the table of 79b, and record in the appropriate column (D = 0 or D = 1) the points to which W, X, Y and Z are connected for these conditions. This time, only points X and Z are equivalent, so, just as we did earlier, we connect these two points together with a curved line and eliminate from either point-X or point-Z any D-contacts leading to the right. This results in the network of 78c.

Looking back through relay-C's contacts from this curved line's ends, to points P, Q and R, we find we can't create any equivalent points from these three, because the three NO-contacts of C lead to three completely different points, which would, of course, make the "C = 1" column of any new table quite incompatible.

And so 78c remains our final non-redundant network, the net saving being one transfer on each of relays D and E.

COMPLEMENTATION OF SYMMETRICAL CIRCUITS

Sometimes it pays to study the specs a little before jumping in and drawing the required network. For example, suppose we were called on to design a circuit for $S^4_{0,1,3,4}$ ABCD. Here we can't readily use the shift-down technique as the subscript progression is irregular, so how about drawing the complement of the desired network, and then, if it's planar (ie, if we haven't used shift-down), we can graphically complement it. In this case, being the naturally observant fellows we are, it strikes us immediately that

$$S^4_{0,1,3,4}\ ABCD = (S^4_2\ ABCD)'$$

and so we simply draw a S^4_2 ABCD network as in Diagram 80a, and then complement it using graphical complementation to produce the network of 80b. 2-out-of-4 is, of course, the hindering-function of 0,1,3,4-out-of-4, which, when complemented, becomes equivalent to the original function.

Note that even during the final process the transfer-contact configuration is preserved.

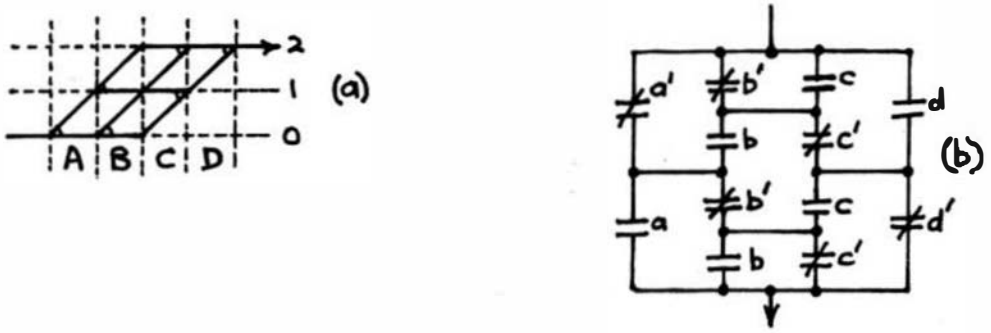


Diagram 80

You should try this yourself, and verify that you can "square up" your final complemented network to give that of 80b.

COMPLEMENTATION OF THE VARIABLES OF SYMMETRY

Here's another extremely useful technique, which may enable a shift-down type of circuit to be produced where it wouldn't otherwise be possible. Let's take another look at Diagram 78, where we were asked to design a $S^5_1,3$ ABCDE.

Now $S^5_1,3$ ABCDE = $S^5_2,4$ A'B'C'D'E'. Remember? (1 relay operated is the same as 4 NOT operated, and 3 operated is the same 2 NOT operated). While a shift-down isn't possible with the left-hand side of the equation (because the next number in the subscript sequence is NOT greater than the superscript) it IS possible with the right-hand side. And so this circuit is designed instead of the original, WITH ONE IMPORTANT DIFFERENCE! The relay designations at the foot of the columns must appear in their complemented form, to indicate that we're now thinking in terms of DE-energised relays. The meaning of the lines is also complemented, HORIZONTAL lines becoming NO-contacts, and DIAGONAL lines, whether normal or shift-down type, becoming NC-contacts.

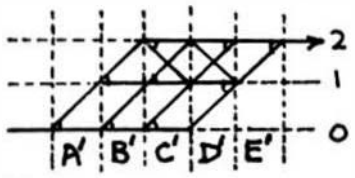


Diagram 81

Diagram 81 shows the network, and I think you'll agree it's a pretty neat one, which took no time at all to design. Let me emphasise, though, that it's VERY important at this stage for you to get in the habit of entering the complement-signs over the variables just as soon as possible, because in the not-too-distant future we'll be dealing with MIXED variables of symmetry. That is, SOME of them will be complemented and others not, and it would NEVER do to omit one of those little signs!

SHIFTING-DOWN WITH THREE OR MORE IRREGULAR SUBSCRIPTS

When three or more subscripts do NOT form an arithmetical progression, it may, or may not, be possible to use shift-down. For instance, Diagram 82a shows an attempt to shift down for $S^5_0,2,5$ ABCDE, by making level-2 a 2,5-level. Unfortunately, it introduces a false output for 3-out-of-5 on level-0.

A better approach would be to slant up to level-3 first, and then shift-down to level-1 to make it a 1,4-level, with a correct output for 5 relays at level-2, now the 2,5-level.

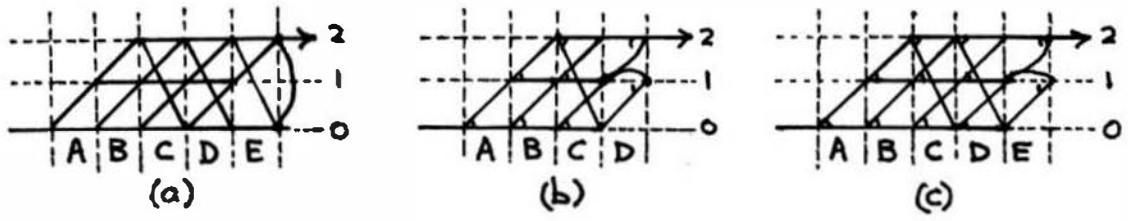


Diagram 82

Very often, the number of variables of symmetry determines whether a shift-down is possible. 82b shows how a shift-down can be made with S^{4}_{1},2,4 ABCD, but is impossible with S^{5}_{1},2,4 ABCDE in 82c. Can you see why? I'll explain, AFTER YOU'VE HAD A GO AT FIGURING IT OUT FOR YOURSELF. We get an output at level-1 for one relay, at level-2 for two relays, then back down to level-0 (with no output) for three. Four relays move us back up to level-1 again, with an output, but we're beaten by the fact that five relays move us up to level-2, still giving an output when none is called for with this combination!

Note how we "wedged" out a transfer in both 82b and 82c, because level-1 and level-2 are one subscript apart!!

Before setting up camp for the night, let me add that, in all the examples we've done, the various output levels could just as well have been separate single outputs, but then we wouldn't be apply redundancy techniques to the networks! Perhaps we could hook this into a PIA, each output level going to a different PIA input. By monitoring the PIA (or perhaps by means of a generated interrupt) our computer would be aware of the ongoing status level of the relays in our network. The computer itself wouldn't have to do any computing to figure out what level we're at, as our "intelligent" relay network would respond almost instantaneously, shifting the levels up and down, and OFF or ON, as the relay-conditions changed. In addition to using these levels as PIA inputs, we could still have them actuate lights, allowing us to visually monitor the network status as well.

Or, of course, we could simply hook the compound-output of our examples into one PIA line, so the computer would know right away, without any actual checking on its part, when the subscript conditions have been met!

That's the end of our arsenal of techniques for tackling symmetric circuit design, but don't go away yet. There's still a lot more to learn about these little beauties, as you'll find out next time!

Anyway, here's a repeat of the last mile's exercises as an official, and expanded-in-scope TEST. You'll enjoy doing these, I know!

TEST TWELVE

1. Design networks, in as many different ways as you can, for the following symmetric functions.

(a) S^{5}_{0},2,5 ABCDE (b) S^{8}_{3},4,5,6 ABCDEFGH (c) S^{6}_{1},4 ABCDEF
(d) S^{7}_{1},4 ABCD (e) S^{12}_{3},10 ABCDEFGHIJKL (f) S^{6}_{1},2,4,5 ABCDEF

I just can't resist telling you how amazed you're going to be next month at the surprising way we're going to shift our viewpoint of symmetric functions, so we can create circuits even more fantastic than those of the last few miles. What a lucky bunch of guys you are!!

... End of Mile 15. Now at marker "Mile 16", but still a long way from journey's end!

SOFTWARE USER NOTES

A Tutorial Series

By : Ronald W Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

From Basic Assembler to HLL's

SK*DOS Advances

I have some most interesting news regarding a new feature of SK*DOS. Peter Stark has implemented sub-directories. For those of us with hard disks, this is a real breakthrough. Peter has not implemented multi-level directories, but he has enabled us to split a drive into a root directory and 26 or 52 sub-directories. The sub-directories are called A through Z. If you enable the flag that turns on the distinction between upper and lower case in filenames, you can also have directories a to z. I presently am using only upper case, so I have the possibility of 26. I haven't had the new version of SK*DOS long enough to be completely at ease with all the new features, but I did manage to modify PAT so it works nicely with the new features. Most utilities automatically work with these multiple directories, but PAT had to save filenames and change extensions, so I had to do a bit of rework to get everything working together.

I can mention a number of uses for the multiple directories. If you have one floppy drive and one hard drive on your system as I do,and you want to copy a floppy to another one, you can set up a new and empty directory as the current directory, and copy a whole floppy into it. Then you can format a new disk and copy the whole directory to the floppy. Of course you can use FTOH and HTOF as well. Then there is the ease of backing up the hard disk, particularly if no directory exceeds the capacity of one floppy. (Every time I have tried to use FTOH to put a whole disk on the hard disk temporarily and they copy it to another floppy (to duplicate a floppy) I have had an error occur, and so have had to copy the disk a file at a time). I suppose my floppy drive is unreliable in some way, but copying a file at a time always seems to work.

The single letter directory names are rather useful. T for text or letter files, A for assembler source files, C for "C" source files, W for Whimsical source files, M for memo files, B for BASIC files, etc. Of course I am never satisfied with what I get, and I've been running MS-DOS and OS9 for a while, so I implemented a CHD command to change the working directory without changing the working disk. CHD T, and whatever directory you are in, you are moved to the T/ directory on the same disk. CHD .. will get you back to the root directory (shades of both MS-DOS and OS9). Borrowing further from OS9, PD will report the current working directory. I also implemented CHX to change the system (execution) directory. Those of you who use IBM compatibles all day might wish to call CHD CD to be more consistent. I'll publish those utilities here in a couple of months, after any bugs in the new directory implementation are sorted out.

Presently, I have the hard disk partitioned as two logical drives, 0 and 1. I use 0 as my "system drive" with all the utilities on it, and 1, which I made twice as large, as the working drive. I don't see much need to fragment the .COM files into numerous directories. They are all software to be run

at various times. Perhaps if a complex compiler came along, with a lot of files, I'd be tempted to put it in its own directory for a while, but otherwise the system files can all be in one.

CSC C Compiler Completed

I recently received a copy of the new C compiler from CSC, ported and adapted by Sid Thompson and Bud Pass. The documentation is an inch and a half thick pile of 8 1/2 by 11 pages, and I have not waded through all of it yet, but I am impressed by the compiler. It is very easy to use and it generates assembler source code to be used by a modified version of the CSC Assembler ASM. The compiler produces absolute code, but it is ORGed far above SK*DOS and I have had no problems in that regard. The libraries are in assembler source code form and it would seem that the compiler automatically pulls in the assembler code for each called library routine. There is presently no linker for this compiler, so that it must compile a program as one unit. The compiler is quick, running substantially faster than the subsequent assembler pass on the compiler output.

A number of demonstration programs were supplied for my testing of the compiler. I haven't explored all of them just yet, but there is a paged listing program, a pretty printing and formatting program for C source code, a couple of nice calendar printing programs and a large number of others. So far, all have compiled and run successfully.

Emacs is an editor written in C, also available from CSC. Sid has ported the Micro version to the SK*DOS environment, and it runs nicely and has all the necessary features such as a smooth word wrap mode for entering text. It is quite configurable and has key macro capabilities. It can edit multiple files at the same time, split a screen showing portions of two different files in the two screen windows, allow you to move text from one file to another, and a number of other nice features generally found in some of the most advanced editors in the IBM compatible realm. There is a long tutorial file through which you can work while running Emacs, and a file that prints out a two-page summary of the commands.

I was quite impressed with the speed of Emacs, though quite a bit less favorably impressed with the choice of keys. For example, anyone who has used Wordstar with the control keys E for cursor up, D for right, S for left and X for down, (or E, F, S, and C for the same in PAT or PIE) would think the key assignments on a mnemonic basis to be extremely inconvenient. Control keys are P(revious) for cursor up, N(ext) for cursor down, B(ack) for cursor left, and F(orward) for cursor right. If you have a PT68K-2, and you have the IBM keyboard and mono-graphics card, Sid has provided a driver called HERC, that allows you to use the cursor control keys on the IBM keyboard for cursor manipulation, and that is of course satisfactory and desirable for those who use IBM compatibles some of the time, and have become accustomed to using the cursor keys.

All is not lost even if you don't like the key assignments for Emacs, however, since keys may be reassigned by means of key reassignment control character sequences. I asked Sid if I could set up a key assignment file and get Emacs to read it and configure itself on startup, and he indicated that it may be possible. I am awaiting further information from him, but meanwhile I can reassign keys and experiment for a while.

Do I like Emacs better than my own creation, PAT? Of course not. As I pointed out to someone recently, If I found something I liked better, I'd simply change my PAT to do what I liked better. As I say whenever I discuss editors, they are probably the most personal software that a programmer uses. There is plenty of room for varying tastes and different editors. The fact that this one may be personalized rather heavily, makes it much more suitable for a wide range of tastes than it would otherwise be. Actually I found the use of more or less mnemonic keys in the less frequently used commands to be a large help. I would only like to change the cursor motion keys to a set that are significant for their position on the keyboard rather than for the mnemonic value. My only

other comment regarding Emacs is that the .COM file is over 100K of object code. So what? The editor is probably the most used single program that any of us owns. Half a percent of our 20 Mbyte hard disk storage space is a tiny price to pay for a very capable editor.

I might also add that in my brief exposure to Emacs, editing a file or two a few nights ago, I found without doubt that I liked it better than just about ANY of the IBM compatible editors that I have tried. That list includes PC-Type, PC-Write, Kedit, PQFedit, and the better of the two offered by Paperback Software, tha name of which escapes me presently, because I haven't used it in so long. Emacs, unlike most of the IBM compatible offerings, does NOT clutter the screen up with endless smiling faces and spade, heart, club symbols all supposed to have significance. It doesn't display "soft hyphens" and "soft carriage returns" as do many of the IBM compatible editors. There is normally just a single status line at the bottom of the screen, though you can ask for and display much other information temporarily. Enough said about Emacs except to tell you to try it, you might like it a great deal. I understand from Sid that there might be a version available for the IBM compatibles so you could edit on your daytime IBM clone and your night time 68020 system with the same identical editor! I'll have to check with Sid regarding how to get that MS-DOS version.

Now, back to the compiler. The C compiler, as I said before is quite acceptably fast, many times faster than any C compiler I ever ran on a 6809 system. It implements virtually all of the features in K&R, (Kernighan and Ritchie, the Bible of the "C" language) other than data type float and double, and the corresponding floating point arithmetic functions. Floating point is not needed (hardly ever, anyway) in writing system programs such as text file filters etc. I had a discussion with Sid some time ago regarding floating point arithmetic. He said that he had written a large number of programs (he does that for a living) in the past ten years, and that about two of them had needed floating point arithmetic. I responded that I too had written a large number of programs over that same time period, and there were two or three that didn't need floating point arithmetic. No, one of us is not wrong and the other right, it is just that our applications are in different realms. I am usually trying to find the volume of the material removed by drilling a hole to the depth of the conical drill tip, or perhaps by milling a flat on the outside edge of a round part, etc. If I am not doing that, I might be doing some vector manipulation, finding the volume of liquid in a cylindrical tank laid flat based on the measurement of the depth of the liquid in the tank and its dimensions, etc. Sid is more involved in the systems programming aspect of computing, (In my mind, at least, more difficult than the sorts of things that I do with computers). He makes different computers talk to each other by means of modem programs (CMODEM is one of his efforts). Notwithstanding Sid's applications, I would certainly be more than pleased if he and Bud would someday consider adding at least floats to their C. I can do without double precision floating point for about 99% of my applications, but lack of floats limits my use of this C to personal projects in the area of file manipulation. I realize that I am in the minority in this wish, and lack of floating point should not deter anyone from purchasing this C who doesn't need those capabilities.

General Observations

The list of available software for SK*DOS has grown substantially in recent days. There are now five editors, Peter Stark's Edlin, (a simple but effective line editor), ED, a reasonable screen editor, Pat, (a little more capable in my opinion), and Micro Emacs, a rather full featured and well written screen editor, and EDDY from Palm Beach Software. Each has its place. Then there is the C compiler, a soon to be ready BASIC, Whimsical (probably to be more or less complete by the end of this year), and UBASIC which has floating point capability but is limited in other areas. There are two assemblers, ASM from CSC and ASMK from Palm Beach Software. SK*DOS itself is coming along with new features such as the multiple directory facility described above. Perhaps

the thing has reached "critical mass" and a chain reaction will take place, more people using SK*DOS inspiring more software so that there will be more people interested in using it, etc. etc.

If things settle down at work in the near future, and Summer ends so things get back to normal activities again, I intend to do some in depth looking at some OS9 software now that the company has the latest update version for the -020 system. There is the latest and greatest OmegaSoft compiler to look at, and some things that Frank Hoffman of Lloyd I/O has said that he would send for evaluation. Then there's those disk transfer utilities from Granite Software that I said I would look at months ago before I found out that it wouldn't run with the older version of OS9. Perhaps I can get the new version installed soon and get on with those projects.

More On C

I have had further time and opportunity to use the C compiler mentioned above. First, I ported over my JUST.C from OS9. I had made JUST (a text formatter) run under Microware C, and was interested in finding out if I could compile it with the new compiler in SK*DOS. To my amazement, the file compiled and assembled WITHOUT ERROR! It didn't quite run on first try, however, and I found a single error in the new compiler, a failure of the feof() function which is supposed to return zero if a file is not ended, and non-zero at the end of the file. It turns out that most "C" programmers use something like the following to read and process a file:

```
while (ch = getc(infile)!= EOF)   {     process a character   }
```

I had used a more conventional Pascal structure since the program had been written in Pascal initially and translated later:

```
while (feof(infile) ==0)   {     ch = getc(infile);     process a character   }
```

Sid Thompson has since found and fixed the problem with the feof() function and returned a working version of it to me. The other problem that I encountered has to do with how C works in various operating systems. In OS9, FLEX, and SK*DOS, a text file has end of line signaled by a CR. In other operating systems such as CPM and MS-DOS, the condition is signaled by LF. Without going into great detail, when I compared the character from the file with '\n', the "newline" character, the test worked fine. I had compared with a CR in the OS9 version so I had to make the change. The comparison with \n will work in both (and probably all) versions of C. It would seem that the test that I had chosen was simply not the most portable one. JUST now runs fine in the new C. ,p Somewhere along the line several years ago, Lane Lester modified my original JUST to work with a proportional spacing printer, and had given me a copy. I started at that and did a C version, and it too runs in the new C. This time, I made the necessary change to compare with \n, and with the repaired feof(), the program not only compiled first try, but it ran also.

A letter from Sid indicates that the next project in expanding the C compiler will be to add floating point capabilities. Of course I wait expectantly for that addition.

Whimsical Version 1.3

I received an update on Whimsical a couple of weeks ago. This latest has floating point capabilities. I spent a week of evenings writing improved scientific function procedures for it, and all are now working. The functions run fairly fast, though I haven't done extensive timing tests on all of them, the square root, for example on the PT68K-2, runs just about 800 microseconds regardless of the input value. The arithmetic package uses a 24 bit 2's complement mantissa and an 8 bit 2's complement with 127 offset exponent. The package takes advantage of the fact that a normalized floating point number always has a 1 in the leftmost bit, by not including that 1 in the number representation (or if you like, hiding the 1 under the sign bit). The extra bit of resolution gained by this technique means that a full 7 decimal digits are available. If this technique is not used the largest number

that can be represented is just over 8000000, so that a full 7 digits would not be possible without the trick of hiding the high order 1.

I must mention that part of the success with the scientific function package is that there is sufficient mechanism in Whimsical to allow direct "bit manipulation" (what Nigel Bennee of Lucidata once called "bit fiddling") on the floating point number representaton. It is possible to get the exponent of the number and divide it by 2, for example, as the first step in getting a good approximation for the first guess in the square root routine.

I have been ironing out the last problems in PAT running on the PT68K-2 with the IBM style keyboard and Monochrome Graphics board and monitor. As soon as I am certain it all works together, we will be offering it for sale. I've resisted the temptation to distribute it until I am sure there are no known bugs in it. That is no guarantee that a few more will not be found by users who try features that I don't normally use, but it does minimize the hassle once a number of copies have been shipped.

RBASIC Arrives

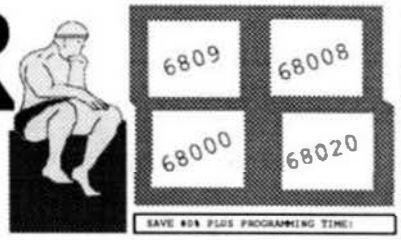
Bob Jones just sent me a copy of RBASIC for the 68000 (and a version for 6809 FLEX but that is another story). I haven't had a real chance to do extensive testing just yet, but I printed out all 64 pages of the manual and read through it once from cover to cover. It appears that RBASIC is completely compatible with TSC's XBASIC. THANKS BOB!! I've moved a few code files over into SK*DOS and they have run without glitches in RBASIC. Bob did TSC one byte better with his REAL arithmetic. The package uses 8 bytes for the number and one byte for the two's complement exponent for a total of 9 bytes. The extra 8 bits of the representation of the number brings the precision to just about 20 digits. Considering the precision of the numbers that may be represented, the code executes very fast. I'll do some hard numbers compared to the old TSC XBASIC running on a 6809 and we'll report the speed increase here perhaps next time when I get to do a full review of the package.

Meanwhile, let me say that the package is impressive. The BASIC is written in Assembler and the object code is under 32K, which is very small for so capable a program. I found the scientific functions to be accurate to some 14 to 16 digits, which is more than enough for just about any application that requires them. In our company applications, the input data is almost never good to more than four places absolute tops, so 6 or 7 digit arithmetic and scientific functions are more than adequate for our calculations without reducing the accuracy of our results. That is, the input data contributes by far the largest factor to the overall error of the calculations. I have always used Extended BASIC to test any new procedures for calculating scientific functions. Now I have a standard in RBASIC for new functions done on the SK*DOS 68000 system.

I ought to report that Bob's 20 digit value of PI does not agree with another source that I have found, after about the 15th digit. I'll send my value to Bob and/or see if its use will improve the accuracy of the functions a little. My first quick test didn't seem to indicate any improvement.

+++

SCULPTOR



From the world's oldest & largest OS-9 software house!

CUTS PROGRAMMING TIME UP TO 80%

6809/68000-68030 Save 70%

SCULPTOR-a 4GL - Only from S.E. Media at these prices. OS-9 levels one and two (three GIMIX) 6809, all 68XXX OS-9 standard systems. Regular SCULPTOR versions 1.4:6. One of if not the most efficient and easy to develope DBMS type systems running under OS-9! A system of flexible keyed file access that allows extremely fast record and data retrieval, insertion and deletion or other programmed modifications. Access by key or in ascending order, very fast. The system provides automatic menu generation, compilation and report generation. Practically unlimited custom input format and report formatting. A rich set of maintenance and repair utilities. An extremely efficient development environment that cuts most programming approximately 80% in development and debugging! Portable, at source level, to MS-DOS, UNIX and many other languages and systems.

Standard Version: 1.6 6809 - $1295.00
68000 $1295.00
68020 $1990.00

Due to a "Special One Time" Purchase, We Are Making This Savings Offer. Quantities Limited!

Once this supply is gone - the price goes back up!

System OS-9: 6809/68000-68030

• Regular ~~$1295.00~~

+ $7.50 S&H USA
Overseas - Shipped Air Mail Collect

ONLY $295.00

S.E. MEDIA
POB 849
5900 CASSANDRA SMITH ROAD
HIXSON, TN 37343 615 842-4601

$AVE - WHILE SUPPLIES LAST!

source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p ,u ,y etc.) With all JUST functions plus several additional printer formating functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

Disk (1) - PL9 FLEX only- F, S & CCF - $49.95

Disk Set (2) - F, S & CCF & OS9 (C version) - $69.95

OS-9 68K000 complete with Source - $79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features and much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

*Regular FLEX, SK*DOS $129.50*

** SPECIAL INTRODUCTION OFFER * $79.95*

SPECIAL PAT/JUST COMBO (w/source)

*FLEX, SK*DOS $99.95*

OS-9 68K Version $229.00

SPECIAL PAT/JUST COMBO 68K $249.00

Note: *JUST in "C" source available for OS-9*

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassel' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - appx. 14,000 plus of free memory! Extra fine for programming as well as text.

*FLEX, SK*DOS $69.95*

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entire lines, etc. Complete with over 25 different CRT terminal configuration overlays.

*FLEX, CCF, SK*DOS $39.95*

SCREDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystrokemacros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-ups", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

*6800 or 6809 FLEX, SK*DOS or SSB DOS, OS-9 - $175.00*

SPELLB "Computer Dictionary" from S.E. Media -- **OVER 150,000 words!** *Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK*DOS UCS).* Or check and update the Text after entry; ADD WORDS to the Dictionary, "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

F, S and CCF - $129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK*DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - $99.95,

F, S or O - $179.95, U - $299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

NEW PRICES 6809 CCF and CCO - $69.95,

F, S or O - $99.95, U - $149.95

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES 6809 CCF and CCO - $59.95,

F, S or O - $79.95, U - $129.95

STYLO-PAK --- Graph + Spell + Merge Package Deal!!!

F, S or O - $329.95, U - $549.95

O, 68000 $695.00

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems
FOR 6809 FLEX-SK*DOS(5/8")

Up to 32 groups/fields per record! Up to 12 character file names! Up to 1024 byte records! User defined screen and print control! Process files! Fonn files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in textline editor! Fully session oriented! Enhanced forms! Boldface, Double width, Italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV

ECHO echos to either screen or file.
FIND an improved *find* command with "*pattern*" matching and wildcards. Very useful.
HEX dumps files in both hex and ASCII.
INCLUDE a file copy program that will accept "includes" of other disk files.
KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.
LISTDIR a directory listing program. Not super, but better than CAT.
MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.
MULTICOL width of page, number of columns may be specified. A MUST!
PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.
REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!
SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.
SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on n° word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.
TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.
TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.
UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.
WC a word count utility. Can count words, characters or lines.
NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, w/ source (PL9). 3 5-1/4" disks or 1 8" disk w/o source.
Complete set SPECIAL INTRO PRICE:
5-1/4" w/source FLEX - SK*DOS - $129.95
w/o source - $79.95
8" w/source - $79.95 - w/o source $49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants - - TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.
F, S and CCF, U - $25.00, w/ Source - $50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/ Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a map! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No 'blind' debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!
Levels I & II only - OS-9 $69.95

DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.
Level I OS-9 obj. $79.95; w/ Source $149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes: REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK*DOS Format so it can be used normally by FLEX, SK*DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK*DOS Directory, Delete FLEX, SK*DOS Files, Copy both directions, etc. FLEX, SK*DOS users use the special disk just like any other FLEX, SK*DOS disk
O - 6809/68000 $79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.
OS-9 $85.00

HIER from S.E. Media - *HIER is a modern hierarchal storage system for users under FLEX, SK*DOS.* It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. **Using HIER a regular (any) FLEX, SK*DOS disk (8 - 5 - hard disk) can have sub directories.** By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. **Each directory looks to FLEX, SK*DOS like a regular file,** except they have the extension '.DIR'. A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK*DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!
*FLEX - SK*DOS $79.95*

COPYMULT from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK*DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.
*Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, SK*DOS, 8" or 5") $99.50*

FORTH

A Tutorial Series

By: R. D. Lurie
9 Linda Street
Leominister, MA 01453

HASHING

Hashing is such an important part of computer programming that it is surprising how little one finds of it in the popular computer press. Usually, if you want to know what hashing is or how to use it, you have to go to some rather esoteric books on computer programming theory. Maybe, if you study computer programming in college, you have so much exposure to hashing that you tend to think that every body is sick of the subject. Off hand, the only other places that I can recall seeing anything significant on hashing was one article in "Kilobaud" and one in "Dr. Dobb's Journal." Since I never studied computer programming in college (I majored in chemisry in 1952-1957), my limited information on hashing has come from those two articles or from books that I could get the local library to borrow for me.

My particular interest in hashing at this time is related to my interest in writing an efficient spelling checker. I have one which I wrote in assembly language a couple of years ago, and it works well enough, but it is somewhat awkward to use. I also made the mistake of trying to build a dictionary which was too general for my type of writing. I started out the dictionary by simply typing in words from the old (book) dictionary which I had been using for spelling references, without particular regard as to what the words were. As a result, the dictionary became cluttered with words which I never used, but which slowed down the search for a word.

Since the purpose of hashing is to generate a single number which is representative of an entire word, it should be possible to find a word among a group of words, if they are somehow keyed by the hashing value. Even though there is an infinite number of possible hashing algorithms, the same word would always give the same hashing value if one only used the same algorithm to find a word as was used to store the word in the first place. This latter fact should be obvious, but it is easy to forget in the heat of programming; as a result, you may change the hashing algorithm, but forget to change the group of hashed words you are using as a testbed. Obviously, this will cause all kinds of irritation while you try to figure out why the system suddenly no longer works!

Hashing cannot provide a completely unique number to describe a word, so that you will eventually find a clash in values; this is known as a "collision." Once a collision occurs, some other way must be found to separate the words.

The analogy that I like is that a hashing function separates a group of words into "buckets" of words, each of which has the same hashing value. If you could have enough buckets, there would never be a collision!

The more complex the hashing function, the longer it takes to run, but it may not necessarily produce any better hashed values. Like most everything else in life, there is a trade-off; in this case one trades other factors for speed, and, possibly, reduced memory consumption. In my opinion, one must accept that collisions will occur and plan for them, since there cannot be an

infinite number of buckets. It seems to me that the ideal hashing function will provide a uniform distribution of words across the span of buckets, so that there will be a minimum of bias introduced into what should be a random search.

I think that the choice of the number of buckets is critical, but not necessarily obvious. When an opinion was given, the concensus appears to be that the number of buckets should be a prime number. I don't understand why, but I am willing to accept the statement: does it insure that there will never be a bucket #0? In all of my experiments with various hashing algorithms, I have never had a case of a hashing value of 0. Is that just a coincidence?

Just think, at last we finally have a practical use for the "sieve" benchmark!
The only real problem that I can see with the need for a prime number is that it is necessary to divide by this number in order to set the number of buckets. If you could use just any number, you could use simple bit-shifts to do the division and, thereby, gain a significant speed advantage. Using a prime forces you into a conventional division operation. In any case, the actual bucket selected is the remainder after the division.

A Practical Example

A article in "Computer Language" for November, 1986, by Dave Taylor contained a table of the 100 most common words in a million-word sample. I am not sure of the source text used for generating this list, so I don't know how closely it fits my writing style, but a cursory examination of Taylor's table looks like a pretty good first-approximation. Table 1 summarizes the size of the words in the experiment and the number of each size.

Number of letters/word	Number of occurances
1	2
2	20
3	30
4	34
5	12
6	1
7	1
total:	100

Table 1. Number of words of each length in the sample tested.

Obviously, the two 1-letter words are "a" and "I." The six-letter word is "before" and the seven-letterword is "through." The rest of the words are pretty much what you would expect. I used the data in his table and this month's FORTH program to generate the data in Table 2.

Number of words per bucket	Number of occurances
1	39
2	16
3	7
4	2
total:	100

Table 2. The result of hashing the 100 most common words.

The results are certainly instructive. Only 64 of the possible 313 buckets were occupied (about 20%). Of these 64 buckets, only 2 (3%) contained as many as four words. The vast majority (61%) of the 64 buckets which were occupied contained only one word. At this stage, at least, there is no evidence requiring that the algorithm be rejected; in fact, it looks good enough to me so that I will continue to use it for the rest of the program development. I may have to change it later, but I plan to stick with it for now.

The Algorithm

The algorithm is really quite simple. Each letter of the word is converted to its upper-case ASCII value and added to a running total. This sum is multiplied by the number of characters in the word, and the product is divided by a prime number. The remainder from this division is used as the hash value.

My current plan is to use this program, and the resulting dictionary, with all of my FORTHs. This means that I will be limited to a maximum of 315 sectors when I am using Stearns' COLOR-FORTH. I think that there will be enough room on the disk for the dictionary if I assign a single screen of 512 bytes to be a bucket. This means that I will not be able to use all of the capacity of an FF9 disk on a DSDD80 drive, but I will give it a try.

FORTH can treat a screen as a random-access record; therefore, there is hardly a more convenient way to to handle the data. Since, as I have said, I have set an arbitrary limit of 315 screens, or

records. I must select a divisor less than that. The largest prime number less than 315 is, conveniently, 313, so very little of the disk will be wasted. I will talk about the rest of the program at another time. Right now, I want to discuss the FORTH specifics.

Hashing In FORTH

The three screens #1-3 show just the minimum necessary definitions in order to test and experiment with the algorithm. I will present the screen in the form I actually used in a later column.

I originally wrote these definitions without variables; it is really not too much trouble to keep the hash value and word length on the Data Stack, if you want to, since these definitions are so short. However, you really do need to have many copies of these two values readily available if you plan to do much more with the hash value. Therefore, I decided to go ahead and start off with the variables in order to save some rewriting later.

This FORTH program is most easily described by starting at screen #3. This is simply a definition which reads one word into the PAD at line 1. If you are using COLOR-FORTH, you will have to use the definition for EXPECT from my last column; you already have EXPECT as part of FF9; you will have to determine what to do with any other FORTH.

The second line determins the length of the word just entered at the keyboard by using a variation on the FIND-FIRST-BL definition from my last column. FF9, or any other FORTH-83, already has this value in the USER variable called SPAN . so you can read that instead of running FIND-FIRST-NUL . In any case, you must store the character count into VLENGTH at this point.

Line 3 does the actual calculation of the hash value and stores the result into VHASH .

The remainder of this definition simply prints the results of the calculation in a convenient form for review.

FIND-FIRST-NUL may not be necessary for your FORTH, but I included it because it will work with any FORTH that I know of. In other words, this definition was included to insure transportability!

The real point of the program is in screen #1, which has the definition of HASH . This definition must be entered with the word length already stored in VLENGTH and the address of the first character of the word on the Data Stack. I think that the comments do an adequate job explaining the purpose of each line, so I won't repeat that here.

The algorithm used is the one stated previously, except that all letters are forced into capitals. The purpose of this is to reduce the size of the sum of the ASCII values, and to insure that "WORD", "Word", and "word" would all produce the same hashed value. I debated whether to force capitals or minuscules, since I wanted one or the other as a way to limit the size of the dictionary.

I finally decided on capitals simply in order to reduce the absolute value of the sum of the ASCII values. Remember that the normal path for FORTH is to use signed math, so that any number greater than 32767 will be treated as a negative number. This is no problem in the math, but it presents the logical problem of having buckets with negative numbers. The obvious solution was as in line #13 with ABS where I simply forced a positive value. This probably has an effect on the randomness of the bucket distribution, but I don't know what it is. Certainly, this is a problem only with very long words, so it is only a kind of last-ditch defense against an unexpected system crash.

TIPS FOR BEGINNERS

There is a hidden bug in screen #1 which you may not see at first! Line #7 will not work for all cases!

The use of the phrase 95 AND was fine as a shortcut for quickly testing the hashing algorithm, but it would never do for the final program. Can you see the problem? I'll save you some trouble by pointing it out. The phrase 95 AND will completely foul up punctuation marks! I want the dictionary to contain words like "can't" and "A/D", and neither of these punctuation marks can get past 95 AND .

The proper way to do this job is shown in screen #4. Remember to test all the possible cases that you can, before committing to a final program.

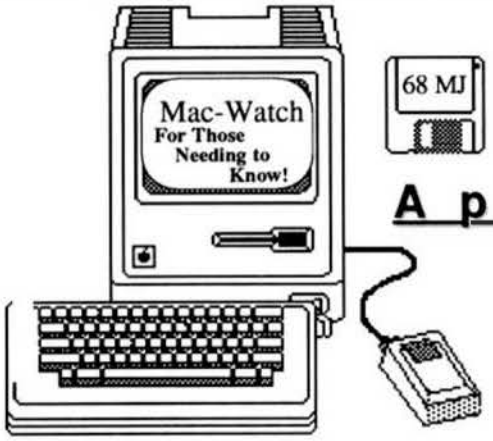
```
*SCR #1
  0 VARIABLE  VHASH              \ hash value
  1 VARIABLE  VLENGTH            \ word length
  2
  3 : HASH  ( adr )              \ RDL080288
  4    0  VHASH  !               \ initialize sum
  5    VLENGTH  @  1+  0  DO     \ set loop limits
  6      ( adr )  DUP  I  +  C@  \ fetch character
  7      95  AND                 \ force upper case
  8      VHASH  +!               \ add to running total
  9    LOOP
 10    DROP                      \ scrap adr.
 11    VLENGTH  @    VHASH  @  *  \ length * sum
 12    313  MOD                  \ divide by prime
 13    ABS                       \ force positive number
 14    VHASH  ! ;                \ save hash value

SCR #2
  0 : FIND-FIRST-NUL ( adr - )  \ RDL080288
  1    -1  VLENGTH  !            \ initialize count
  2    BEGIN
  3      DUP                     \ working copy of adr.
  4      1  VLENGTH  +!          \ incremnt count
  5      VLENGTH  @              \ fetch current count
  6      ( adr count )  + C@     \ fetch current character
  7      0=                      \ is it a NUL?
  8    UNTIL                     \ loop on FALSE flag
  9    DROP ;                    \ scrap adr.

SCR #3
  0 : TEST   ( - )              \ RDL080288
  1    PAD  25  EXPECT          \ fetch input word
  2    PAD  FIND-FIRST-NUL      \ get word length
  3    PAD  HASH                \ get hash value
  4    PAD  VLENGTH  @  TYPE    \ display word
  5    5  SPACES
  6    VLENGTH  ?
  7    VHASH  ?
  8    CR ;

SCR #4
  0 : FORCE-CAPITAL  ( c1 - c2 )  \ RDL080288
  1    DUP    95  >              \ ASCII underline
  2      IF    32  -
  3      ENDIF ;                 \ use THEN for FF9

+++
```



The Macintosh™ Section

Reserved as

A place for your thoughts

And ours.....

Mac-Watch

A Review of Canvas Version 1.02M from Deneba Software

By James E. Law
1806 Rock Bluff Rd.
Hixson TN 37343

Just when you thought you had all the graphics programs you need, along comes another one to tempt you. Magazines flash full-page ads to persuade you to let go of that hard-earned money to obtain those new features that your last purchase didn't have. Canvas, a full featured graphics program from Deneba Software claims to be a better mousetrap. Is it? Let's look at it.

In this review, I will make a number of comparisons to other programs that you are familiar with, such as Super Paint and MacPaint. This is the easiest way I know to explain the program with a minimum of words.

Let's Get Started

Canvas, like SuperPaint and other modern graphics programs, has both bitmap and draw imaging capabilities. Before we get into an analysis of features that are unique to one of these modes, let's look at some capabilities that are common to both modes.

I really like the way Canvas allows you to view your work at various sizes. You can magnify your work up to 32 times regular size for easy editing of super high resolution paint objects. Going the other way, you can view your document at as little as 1/32 of its regular size. This will be essential for working on very large documents. (Canvas allows a work area of 9 ft.by 9 ft.!) This feature is easily and quickly selected and all tools are active in the alternate views. This feature allows full page previews regardless of page size.

Canvas has an adjustable grid which may be used to aid layout. A "snap-to" feature can be used with this grid for precise placement of drawing elements. While the grid can be hidden, you have to do this each time you open a document. (Deneba, why didn't you provide a saveable "preferences" file?) Vertical and horizontal rules are provided with many more features than provided by competing programs. The ruler specification window allows you to set the units to be used (inches, pixels, centimeters, or custom) and the major and minor divisions (e.g., 10 divisions per inch, 16 divisions per inch) An especially nice feature is the ability to "tear away" one or more copies of the rulers and place them wherever you wish on the screen.

Painting the Town

The paint mode of Canvas has a menu very much like other paint programs with lasso, marquee, text, spray can, line, paint brush, paint can, pencil, eraser, and shapes tools. There's a fundamental difference from some other programs, however, in that you can't just select a paint tool and start to work. Instead, you must designate a portion of the screen to be a paint area. Once such an area is designated, you may begin to paint—but only in that area. This is definitely not as flexible an approach as used by several other graphics programs. To handle text, however, Canvas has a leg up on the competition in that it allows

you to go back and edit text you have entered as long as you have not clicked outside the text block.

Double-clicking the paint brush reveals a menu of 40 brush shapes. Double-clicking the paint bucket reveals a menu of 76 fill patterns. Empty boxes are provided so that you can design and save up to 38 additional patterns. Double-clicking the text tool brings to view the text specification dialog box where you can set font, style, font size, and justification. Any text size, up to 127, may be entered. A little faster approach to setting text specs is to hold down the option key while selecting the text tool. A hierarchial menu instantly appears for making your choices of font, style, and size. Using the command key in this way allows you to chose the font.

Canvas allows considerable control of this paint spray can nozzle. You may edit the flow pattern by individual pixel as well as the flow rate (from 0 to 25). At "0", no paint flows except when the cursor is moving while at "25", paint flows freely even while the cursor is still.

Horizontal and vertical pen thickness may be set individually with 10 choices available for each. Pen patterns may be set in any one of 38 patterns.

For those who need to draw flow charts will appreciate the arrow lines. This option automatically terminates any line drawn with an arrow pointing in the designated direction.

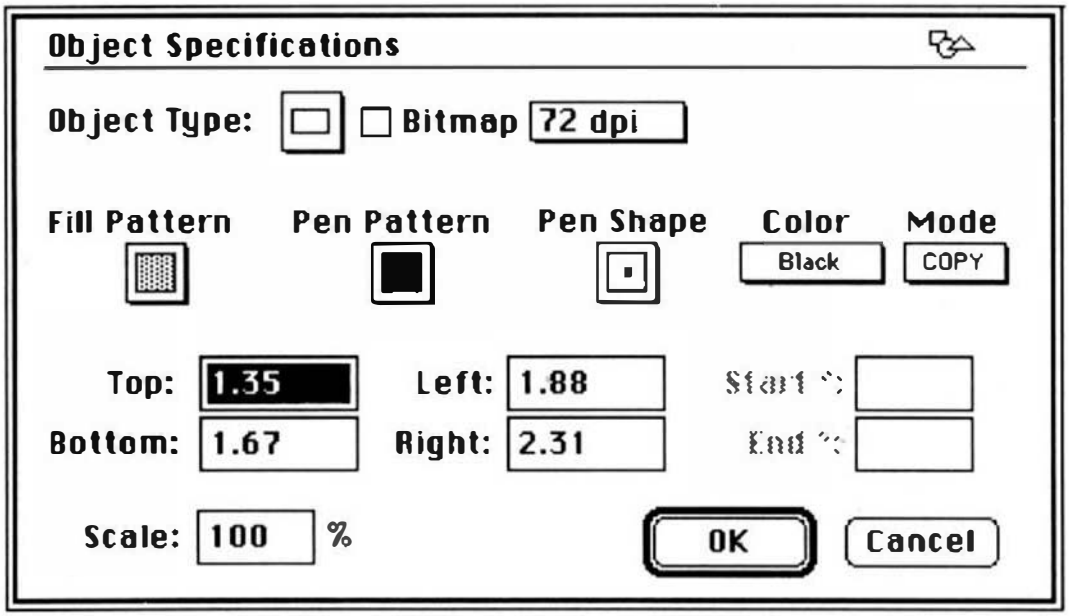
Draw it Again

Most of Canvas' drawing tools are pretty standard. They include geometric shapes (rectangles, circles, rounded corner rectangles), lines, text, free hand shapes, and polygons. Polygons can be edited or smoothed. Also, Canvas allows you to specify one or more control points on a polygon that remain as a sharp vertex or ("cusps"). After the "Smooth Polygon" is selected. (This would be helpful if you were drawing an ice cream cone.)

You may group multiple objects together so that they can be handled as though they were one object. Objects and groups of objects may be "locked" so that they will not be inadvertently moved or deleted. You can, of course, send objects to the back of other objects or bring them to the front.

Objects may be aligned to a line by their top, bottom, sides, or center. Want to make a copy? It's quick and easy. Just select the object and click the copy icon in the tool box. Canvas allows you to get a lot fancier with your copying, however, by selecting "Duplicate Options" from the edit menu. This feature provides an almost unbelievable control of the duplication process. You can specify:

(a) The number of copies
(b) The incremental amount each copy is to be rotated
(c) The horizontal and vertical offset of each copy. (This would be great for drawing forms)
(d) The amount by which each copy will be resized (enlarged or reduced). The horizontal and vertical dimensions can be resized separately.
(e) Each copy can be set to appear in a different pattern
(f) The center of rotation for rotated copies may be specified.



Canvas's "Object Specification" Dialog Box Allows Precise Control of Object Specifications

It didn't take me long to figure out that this is a **very** powerful feature. Truly impressive effects can be achieved with relatively little effort.

This version of Canvas allows benzier curves with 4 control points. A more sophisticated approach is promised for the next version of Canvas.

Those who want the ultimate in precise control of their drawings, the "Object Specification" dialog box will be appreciated. It allows one to set the object type, the resolution of bit map objects (up to 2540 dpi), fill and pen patterns, pen shape and color, exact screen coordinates, and its scale (i.e., it may be enlarged or reduced by the specified percentage).

Canvas provides a collection of special effects for manipulating objects. They may be rotated 90 degrees, free rotated, and flipped horizontally or vertically. Objects can also be modified to show 1 or 2 point perspective, skewed, or distorted. All these features print at the full resolution of you printer and also work with text if you have a post script laser printer. The manual says that with non-post script printers, only a "paint" image of the text will be printed. You can't believe everythingyou read, however. On my Quick Draw laser printer (Personal Laser Writer from General Computers), rotated text printed at full 300 dpi resolution.

Macros

One feature provided by Canvas, but not by most of its competitors, is the ability to create and save draw-format images as "macros." Any draw-format image may be turned into a macro by selecting it and choosing "Add Macro" from the Macro menu. The macro may then be named and recalled when ever needed. This image 'scrapbook' will preclude the need to redraw frequently used objects. This feature might be used by a draftsman to store electric circuit symbols, by an interior decorator to store a chair icon, or by a systems analyst to store standard flow chart symbols.

Canvas DA

One Canvas feature that is head and shoulders above the competition is its desk accessory version of the stand alone application. This 50K DA has about 80% of the features contained in the stand alone applications. Some of the features not included in the DA version include:

(a) Duplicate options
(b) Edit patterns
(c) Edit brushes
(d) Edit spray nozzle
(e) Object specifications dialog box
(f) Smooth/unsmooth polygons
(g) Custom rulers
(h) Grid specifications
(i) Free rotate
(j) Special effects (perspective, skew, distort)
(k) Macro

Don't let this list lead you to conclude that Canvas DA is skimpy on power. Instead, it is probably the most powerful and feature-laden graphics DA currently on the market.

Basic Stuff

Canvas runs on any Macintosh with at least 512K. It's manual is, for the most part, fully adequate. It contains a number of illustrations which make the material easy to understand.

Conclusion

I like Canvas' power and its wealth of special features. I also like the easy to use hierarchial menus. The full-featured DA included for the same money is also a big plus. I do not like the non-intuitive way it handles the designation of bit map and object images. All-in-all, Canvas is a high quality tool for generating graphics. At a street price of as little as $109, you will be getting your money's worth.

Deneba Software has scheduled Version 2.0 of Canvas for release later this year. If you are interested in graphics software, you'll want to watch for this release since a number of new features are promised.

EOF

Add Graphics To your SBC

by Joseph D. Condon
8072 172nd Street W.
Lakeville, MN 55044

With the vast selection and low price of single board computers, it has now become possible to assemble a very powerful system for a reasonable price. Unfortunately, most single board computers lack the integral graphic system found in most commercial PC's.

A few SBC manufacturers offer graphic controls for their systems but they are usually very expensive high performance color controls. In addition to the cost of the control, you have to consider the cost of a video monitor. A good quality high resolution color monitor will set you back at least $500.00. It's apparent that adding graphic capabilities to a SBC will probably be too expensive for a lot of hobbyists.

Because of this void in affordable graphic controls, I decided to design a low cost graphic system that could be used with most SBC's. The general specifications for this system follow.

Cost

Component cost for the control is approximately $50.00. You will also need a monochrome, TTL video monitor. I have seen EGA resolution (640 by 350) monitors advertised for about $120.00. If your system has an extra serial port, you may want to add a mouse device for another $80.00. For a total cost of about $250.00 you can give your SBC complete graphic work station capabilities.

To keep the cost of the control at a minimum, I had to impose some definite limitations on the design. The major limitation is the control will only generate monochromatic displays. This limitation not only keeps the cost of the control down but also saves a great deal on the cost of a video monitor.

Performance

High performance in a graphic control can only be obtained by using a specialized graphic processor. There are many graphic processors available which vary widely in price and performance. The one I chose to use for this project is the NEC 7220.

The NEC 7220 has been around for several years now and its price has slowly dropped to under $10.00. The NEC may not be as sophisticated as some of the newer generation graphic processors but then it doesn't cost several hundred dollars either. Besides, any lack in sophistication can be overcome with software.

One major concern I had with the NEC was performance. The performance of any graphic processor usually boils down to how fast and and how often display memory can be accessed by the graphic processor for drawing or data transfer functions. Since the NEC does not interleave display memory video refresh cycles with processor data access cycles, actual drawing or data transfers can only occur during the blanked portion of the video display. This means the NEC's drawing controller will only be able to draw at a rate approximately equal in this design to 10 usec per pixel. That's fast enough to draw 1000 lines 100 pixels long in one second which is faster than most PC's. Data transfer cycles are limited to approximately 10 usec per 16 bit word which will usually exceed the bandpass of a typical SBC expansion bus controlled by an intelligent driver.

One characteristic I found unusual about the NEC 7220 was the lack of an interrupt request signal. Since most operating system I/O handlers are interrupt driven, I thought using the NEC would be

impractical. Instead, because of the required interface speed, it was faster to by-pass the operating systems I/O handlers altogether, and to do this you don't want an interrupting device. As it turns out, the NEC 7220 fits well into the overall design and will provide more performance than most SBC applications will be able to utilize.

Construction and Implementation

To simplify construction, I have made every effort to keep the component count to a minimum. The total chip count is 20 and this includes the crystal oscillator. The only other components required are bypass capacitors, one pull-up resistor and one filter capacitor. All the necessary components can be purchased from Jameco Electronics or JDR Micro Devices. Be sure to purchase the specification sheets for the NEC 7220. They contain all the information you should ever need to know about the controller.

Actual construction technique and component layout is by no means fixed. The example layout is what I choose to use for my SBC which lives in an IBM XT case. The example layout produces a circuit board that can be mounted in an empty disk drive location and draw its power from the extra disk drive power connector.

The host processor interface cable and connector will be unique for each brand or style of SBC. The pin out of the interface shown in the schematic is unique to my system and will have to be altered to work with a different SBC. As presented, the interface used by the control is standard Motorola 6800/68000. If you choose to connect the control to an Intel processor based system, certain minor changes in the controls interface logic will have to be made. Since the NEC 7220 was designed specifically for the Intel bus, the required changes should be fairly easy and straight forward however, the software drivers will not work on a non memory mapped I/O system.

Display Format and Resolution

The display format of the control is extremely flexible. The format and resolution are determined by the crystal oscillator frequency and the software driver. The following table lists the four basic modes of operation.

Mode	Horz	Vert	Oscillator Jumper	
CGA	640	240	12.0 MHZ	Low
EGA1	640	350	18.432 MHZ	Low or Hi
EGA2	704	350	18.432 MHZ	Low or Hi
VGA	640	480	24.0 MHZ	Hi

If you are adventurous, you can experiment with different values of crystal oscillators. Of course the initialize strings in the software will have to be modified according to the desired display format and resolution.

Caution!!!

You may have heard the statement "software cannot damage hardware". One major exception to that statement is the video monitor. Some monitors tend to leak smoke when driven by improper syncronization signals. If you should have the misfortune of damaging your monitor or computer while using this control, you have my deepest sympathy but nothing more.

Processor and Operating System Independence

The only way to attain processor and operating system independence is to write all of the software, including the drivers in a transportable programming language. The C language is well suited for this task and most SBC's have C compilers available. As implemented, the driver talks directly to the control and the operating system is oblivious to the controls existence. This may be considered improperprocedure by some but it makes for a very quick and flexible interface.

The driver can be used as a library of high level functions easily accessible to user applications. The driver depends upon a header file which contains specific information relative to it's environment. The driver source file should never need to be modified, only the header file. The sample demonstration program shows how the graphic programing environment is used.

How Does It Work?

The NEC 7220 graphic controller is a fairly complex device and to try to cover all of its intricacies would be beyond the scope of this article and my knowledge. Instead, I will try to present a brief overview of how the controller operates in this specific design.

The control occupies two memory address locations on the standard Motorola peripheral bus. Using these two locations, the host computer is able to read the controllers status register, write commands to the controller and read and write data. Since the control can not generate interrupts, all communications with the device must be performed using the controllers status register and a polling type of software driver. This type of interface is quicker than a normal interrupt driven handler but it does require a lot of the host processors resources.

The NEC graphic controller is best described as a special purpose micro processor. It has its own local memory array which it controls by means of a multiplexed address/data bus and a single address latch enable signal (ALE). When ALE initiates a memory cycle, the supporting circuity must provide all of the necessary signals to perform the actual memory cycle. In this design, a simple state machine generates these signals. The state machine is composed of two 8 bit shift registers, several gates and several inverters.

The controller can only perform two types of memory access cycles, read and read modify write. The state machine always performs a read cycle when ALE goes false. At the end of the read cycle, if ALE is still false, the state machine continues on with the modify portion of the read modify write cycle. When ALE is true, the state machine is held in a reset condition.

The controller also provides all of the video control signals necessary to drive a monitor. These control signals are latched by the state machine before being presented to the monitor. The oscillator provides the pixel shift clock and is divided by 2 for the state machine and by 8 for the controller. The jumper selection for low and high resolution operation was necessary to compensate for propagation delays in this design.

The memory array is made up of four 64K by 4 bit memory devices which provide a 16 bit data path for the controller and video shift registers. The memory addresses generated by the controller are multiplexed into the memory devices by the two 74LS257's which are controlled by the state machine. To guarantee operational requirements of the memory devices, the upper half of the address bus must be latched at RAS time by the 74LS273 which is also controlled by the state machine. The two 8 bit video shift registers are connected in series to provide a 16 bit pixel shift register who's output is gated by the video blanking signal and feed directly to the monitor as video data information. The shift registers are loaded from display memory at the end of each read cycle by the, you guessed it, state machine. The state machine is the heart of this design and its economical construction is the key to the controls simplicity and low cost.

The C Driver

Because of the complexity in command format and protocol of the NEC 7220, I decided to make thedriveras intelligent as possible with out sacrificing any of the controllers utility values. The driver directly accesses the controller via short integer pointers. The application program need never directly access the control, it should only call driver procedures which actually perform the desired function. Maintaining this level of hierarchy will help insure portability of the application.

The C driver is composed of many individual procedures which eventually get linked into the application program. The driver requires the presence of a header file which defines several global constants. The application should also require this file during compilation.

An example application program procedural call follows;

```
g_init(VGA);
```

This procedure initializes the control and configures it for the VGA mode of resolution (640 by 480). The variable "VGA" is one of four pre-defined modes of operation defined in the header file.

The controller divides its display memory into two separate windows, the upper and the lower. These two windows operate independent of one another and can be of any reasonable size provided they both share the same width. The two windows can be independently scrolled. Vertical scroll resolution is in pixels and horizontal in words or units of 16 pixels. The upper window will always begin at the top of the display screen and the lower window can be positioned anywhere below the upper. The lower window can fill the entire screen, masking out the upper or it may not even appear on the screen. With the controls 1,048,576 bits of pixel memory, a typical display configuration for an interactive CAD type program might be;

upper = 1200 wide by 800 tall
lower = 1200 wide by 73 tall

Another example of a driver call is the line drawing procedure.

```
g_line(UPPER,REP,&lpt(SOLID),x1,y1,x2,y2);
```

This procedure draws a line in the upper window from coordinate x1,y1 to coordinate x2,y2. The "lpt" variable is a pre-defined array of line pattern images contained in the global file and the constant "SOLID" has been defined in the header file. The constant "REP" has been predefined and indicates that the line is to be drawn in replace mode (replace all pixels with pattern image). The controller has 3 other modes of operation, set, reset and compliment. There are also driver procedures that draw dots, rectangles, circles and rectangular areas filled with pre-defined 8 by 8 bit pixel patterns. The driver also has text related procedures which enable easy character or character string displaying. Text can be magnified up to 16 times, slanted and printed in any one of eight different directions.

There is also a very fast area clear procedure that can clear the entire memory array in less than one second. To use the memory data read and write commands, you will have to study the specification sheets. These functions are very application dependent and require additional logic on the application side of the program.

The demonstration program in addition to the global, header, driver and makefile is a complete graphic programing environment. By studying the listings you should be able to determine exactly how each procedure is used. The driver can be modified or expanded if you like. You may even want to optimize some of the procedures by converting them to machine specific assembler routines. I have written a set of pixel block transfer (PIXBLT) routines in OS9/68000 assembler and they really perform well. The routines actually operate as a type of windowing system that allow many overlapping windows to appear on the screen simultaneously without interfering with one another. If there appears to be sufficient enthusiasm, I may present the routines later on in another article.

Conclusion

Hopefully I have sparked your interest in computer graphics and you are willing to invest a little time and money for a worth while project. I really feel that a good quality graphic system is one of the most important assets of any personal computer system. With the proper software this control can be used to draw schematics, circuit board layouts, charts, tables, diagrams and even format text files. If you would like to see additional articles on software for this control, just send me or "68 Micro Journal" a letter. Till next time...

```
/*=========================*/
/*  Graphic Demo Program  */
/*=========================*/
#include <ctype.h>
#include "graph.h"

/*------*/
/*  Main  */
/*------*/
int main(argc,argv)
int argc;
char *argv[];
{
  int i;

  /* initialize control */
  g_init(VMODE);

  /* define the two display
windows */
  g_dd(UPPER,DHGT-
MHGT,LOWER,MHGT);
  while(TRUE) {

  /*----------*/
  /* Line Display */
  /*----------*/
 g_clr(LOWER,RES,0,0,DWDT,MHGT);
 g_str(LOWER,REP,3,0,2,150,40,"Line
Display");
 g_clr(UPPER,RES,0,0,DWDT,DHGT-
MHGT);
  for(i=0;i<=300;i+=10) {
```

```
                                _line(UPPER,REP,&lpt[SOLID],i,0,300,i);
_line(UPPER,REP,&lpt[SOLID],300,i,300-i,300);
_line(UPPER,REP,&lpt[SOLID],300-i,300,0,300-i);
_line(UPPER,REP,&lpt[SOLID],0,300-i,i,0);
_line(UPPER,REP,&lpt[SOLID],330,i,630,i);
_line(UPPER,REP,&lpt[SOLID],330+i,0,330+i,300);
  }
  sleep(5);

  /*-----------*/
  /* Rectangle Display */
  /*-----------*/
  g_clr(LOWER,RES,0,0,DWDT,MHGT);
  g_str(LOWER,REP,3,0,2,100,40,"Rectangle Display");
  g_clr(UPPER,RES,0,0,DWDT,DHGT-MHGT);
  for(i=10;i<(DHGT-MHGT)/2;i+=10) {

_rct(UPPER,REP,&lpt[((i-10)/10)%16],
        DWDT/2-i,(DHGT-MHGT)/2-i,DWDT/2+i,(DHGT-MHGT)/2+i);
  }
  sleep(5);

  /*----------*/
  /* Circle Display */
  /*----------*/
  g_clr(LOWER,RES,0,0,DWDT,MHGT);
  g_str(LOWER,REP,3,0,2,125,40,"Circle Display");
  g_clr(UPPER,RES,0,0,DWDT,DHGT-MHGT);
  for(i=10;i<(DHGT-MHGT)/2;i+=10) {

_circle(UPPER,REP,&lpt[((i-10)/10)%16],i,DWDT/2,(DHGT-MHGT)/2);
  }
  sleep(5);

  /*------------*/
  /* Fill Pattern Display */
  /*------------*/
  g_clr(LOWER,RES,0,0,DWDT,MHGT);
  g_str(LOWER,REP,3,0,2,50,40,"Fill Pattern Display");
  g_clr(UPPER,RES,0,0,DWDT,DHGT-MHGT);
  for(i=0;i<32;i++) {
_frct(UPPER,REP,&spt[i],
        (i%8)*DWDT/8,(i/8)*(DHGT-MHGT)/4,
        ((i%8)+1)*DWDT/8,((i/8)+1)*(DHGT-MHGT)/4);
  }
  sleep(5);

  /*---------*/
  /* Text Display */
  /*---------*/
  g_clr(LOWER,RES,0,0,DWDT,MHGT);
  g_str(LOWER,REP,3,0,2,125,40,"Text Display");
  g_clr(UPPER,RES,0,0,DWDT,DHGT-MHGT);
  for(i=0;i<64;i++) {
    g_chr(UPPER,REP,0,0,2,i*6,10,i);
    g_chr(UPPER,REP,0,0,2,i*6,20,i+64);
  }
  g_str(UPPER,REP,1,0,2,0, 45,"AaBbCcDdEe");
  g_str(UPPER,REP,2,0,2,0, 80,"AaBbCcDdEe");
  g_str(UPPER,REP,3,0,2,0,120,"AaBbCcDdEe");
  g_str(UPPER,REP,4,0,2,0,170,"AaBbCcDdEe");
  g_str(UPPER,REP,5,0,2,0,230,"AaBbCcDdEe");
  g_str(UPPER,REP,6,0,2,0,290,"AaBbCcDdEe");
  g_str(UPPER,REP,7,0,2,0,360,"AaBbCcDdEe");
  g_str(UPPER,REP,8,0,2,0,440,"AaBbCcDdEe");
  sleep(5);

  }
}
```

+++

```
/*==============================*/
/* Graphic Demo Global Variables */
/*==============================*/
#include <ctype.h>
#include "graph.h"

/*-----------*/
/* Line Pattern Table */
/*    16 Patterns     */
/*-----------*/
int lpt[]={
  0xffff,0x5555,0x1111,0x0101,
  0x0001,0x3333,0x0f0f,0x00ff,
  0x0155,0x1555,0x0fff,0x3fff,
  0x9ff9,0xcff3,0xe7e7,0xf3cf,
};
/*-----------*/
/* Solid Pattern Table */
/*     32 Patterns     */
/*-----------*/
struct spe spt[]={
  0xffff,0xffff,0xffff,0xffff,
  0xaa55,0xaa55,0xaa55,0xaa55,
  0x5500,0x5500,0x5500,0x5500,
  0x1100,0x4400,0x1100,0x4400,

  0x0000,0x0011,0x0000,0x0011,
  0x0000,0x0010,0x0000,0x0001,
  0x0000,0x0000,0x0000,0x0001,
  0x0000,0x0000,0x0000,0x0000,

  0x5555,0x5555,0x5555,0x5555,
  0x1111,0x1111,0x1111,0x1111,
  0x0101,0x0101,0x0101,0x0101,
  0x3333,0x3333,0x3333,0x3333,
```

```
  0x0f0f,0x0f0f,0x0f0f,0x0f0f,
  0x00ff,0x00ff,0x00ff,0x00ff,
  0x0000,0x00ff,0x0000,0x00ff,
  0x0000,0x0000,0x0000,0x00ff,

  0xffff,0x0000,0xffff,0x0000,
  0xffff,0xffff,0x0000,0x0000,
  0xff88,0x8888,0xff88,0x8888,
  0xff80,0x8080,0x8080,0x8080,

  0xffff,0x3333,0xffff,0x3333,
  0xffff,0xffff,0x0f0f,0x0f0f,
  0x5500,0x0100,0x0100,0x0100,
  0x8301,0x0000,0x0000,0x0001,

  0x1122,0x4488,0x1122,0x4488,
  0x0102,0x0408,0x1020,0x4080,
  0x8844,0x2211,0x8844,0x2211,
  0x8040,0x2010,0x0804,0x0201,

  0x8142,0x2418,0x1824,0x4281,
  0x8888,0x888f,0x8888,0x88f8,
  0x413e,0x2222,0x223e,0x4180,
  0x805d,0x2241,0x4141,0x225d,
};

+++

/*=========================================*/
/*                                         */
/*  GRAPHIC "C" Driver Routines            */
/*       for the NEC 7220                  */
/*                                         */
/*      Physical Interface                 */
/*                                         */
/*        by Joe Condon                    */
/*            6/24/88                      */
/*                                         */
/*=========================================*/

#include <ctype.h>
#include "graph.h"

static short int
  *necstat=GRSWP,
  *neccmd=GRFWC,
  *necrfifo=GRFWC,
  *necwfifo=GRSWP;

/*-----------------------*/
/* NEC Status Register Bit Definitions */
/*-----------------------*/
#define LPD
x80
* light pen detect */
#define HBA 0x40
* horizontal blank active */
#define VSA 0x20
* vertical sync active */
#define DMA 0x10
* DMA execute */
#define DIP 0x08
* drawing in progress */
#define FFE 0x04
* fifo empty */
#define FFF 0x02
* fifo full */
#define DRY
x01
data ready */

/*---------*/
/* NEC Commands */
/*---------*/
#define
ESET   0x00          /* reset gdc to idle*/
#define SYNC    0x0e          /* specify display format
*/
#define VSYNC
x6e          /* select master or slave mode */
#define CCHAR   0x4b          /* specify character row
heights */
#define START   0x6b          /* end idle and unblank
display */
#define BCTRL
x0c          /* controls display blanking */
#define ZOOM
x46          /* sepcify zoom factor */
#define CURS
x49          /* set cursor position */
#define PRAM
x70          /* define display area */
#define PITCH
x47          /* specify display width */
#define WDAT
x20          /* write data into display memory */
#define MASK
x4a          /* set mask register */
#define FIGS    0x4c          /* specify drawing
parameters */
#define FIGD    0x6c         /* draw the figure */
#define GCHRD
x68         /* draw graphic character */
#define RDAT
xa0         /* read data from display memory */
#define CURD    0xe0          /* read cursor position
*/
#define LPRD    0xc0          /* read light pen address
*/
#define DMAR    0xa4          /* request DMA
read */
#define DMAW    0x24          /* request DMA
write */

/*-------------*/
/*  Initialize NEC 7220  */
/*-------------*/
/* aw = active words     */
/* vs = vert sync width  */
/* hs = horz sync width  */
/* hfp = horz front prch */
/* hbp = horz back porch */
/* vfp = vert front prch */
/* al = active lines     */
/* vbp = vert back porch */
/*-------------*/
void g_init(aw,vs,hs,hfp,hbp,vfp,al,vbp)
int aw,vs,hs,hfp,hbp,vfp,al,vbp;
{
  *neccmd=RESET;
  wrfifo(0x16);
  wrfifo(aw-2);
  wrfifo(vs<<5|(hs-1));
  wrfifo((hfp-1)<<2|vs>>3);
  wrfifo(hbp-1);
  wrfifo(vfp);
  wrfifo(al);
  wrfifo(vbp<<2|al>>8);
  wrcmd(PITCH);
  wrfifo(WWDT/16);
  wrcmd(VSYNC|1);
  wrcmd(CCHAR);
  wrfifo(0);
  wrfifo(0);
  wrfifo(0);
  wrcmd(ZOOM);
  wrfifo(0);
  wrcmd(START);
}

/*---------------*/
/*       Define Display       */
/*---------------*/
/* uwa = upper window address */
/* uwh = upper window height  */
/* lwa = lower window address */
/* lwh = lower window height  */
/*---------------*/
void g_dd(uwa,uwh,lwa,lwh)
int uwa,uwh,lwa,lwh;
{
  wrcmd(PRAM|0);
```

```
  wrparm(uwa);
  wrfifo(uwh<<4|uwa>>16);
  wrfifo(uwh>>4);
  wrparm(lwa);
  wrfifo(lwh<<4|lwa>>16);
  wrfifo(lwh>>4);
  wrcmd(PITCH);
  wrfifo(WNDT/16);
}

/*-----------*/
/*     Clear Area      */
/*-----------*/
/* sa = screen address */
/* md = mode           */
/* x = x coordinate    */
/* y = y coordinate    */
/* w = area width      */
/* h = area height     */
/*-----------*/
void g_clr(sa,md,x,y,w,h)
int sa,md,x,y,w,h;
{
  int i,m,ww;

  ww=1+(x+w-1)/16-x/16;
  for(i=0;i<ww;i++) {
    m=0xffff;
    if(!i) m&=0xffff<<(x%16);
    if(i==ww-1) m&=0xffff>>(15-(x+w-1)%16);
    g_poscur(sa,x+i*16,y);
    wrcmd(MASK);
    wrparm(m);
    wrcmd(PIGS);
    wrfifo(0);
    wrparm(h-1);
    wrcmd(WDAT|md);
    wrparm(0xffff);
  }
}

/*-----------*/
/*      Draw Dot       */
/*-----------*/
/* sa = screen address */
/* md = mode           */
/* x = x coordinate    */
/* y = y coordinate    */
/*-----------*/
void g_dot(sa,md,x,y)
int sa,md,x,y;
{
  g_poscur(sa,x,y);
  wrcmd(PIGS);
  wrfifo(0);
  wrcmd(WDAT|md);
  wrcmd(PIGD);
}

/*-----------*/
/*      Draw Line      */
/*-----------*/
/* sa = screen address */
/* md = mode           */
/* x0 = x coordinate   */
/* y0 = y coordinate   */
/* x1 = x coordinate   */
/* y1 = y coordinate   */
/*-----------*/
void g_line(sa,md,pat,x0,y0,x1,y1)
int sa,md,*pat,x0,y0,x1,y1;
{
  static int
dtab[]={0x9,0x8,0xa,0xb,0xe,0xf,0xd,0xc};
  int i,xt,yt,dl,ds;

  i=0;
  xt=x1-x0;
  if(xt<=0) {
    xt=abs(xt);
    i+=4;
  }
  yt=y1-y0;
  if(yt<=0) {
    yt=abs(yt);
    i+=2;
  }
  if(xt>yt) {
    dl=xt;
    ds=yt;
  }
  else {
    dl=yt;
    ds=xt;
    i+=1;
  }
  g_poscur(sa,x0,y0);
  wrcmd(PIGS);
  wrfifo(dtab[i]);
  wrparm(dl);
  wrparm(2*ds-dl);
  wrparm(2*(ds-dl));
  wrparm(2*ds);
  wrcmd(PRAM|0x08);
  wrparm(*pat);
  wrcmd(WDAT|md);
  wrcmd(PIGD);
}

/*-----------*/
/*      Draw Circle    */
/*-----------*/
/* sa = screen address */
/* md = mode           */
/* pat = line pattern  */
/* r = circle radius   */
/* x = x coordinate    */
/* y = y coordinate    */
/*-----------*/
void g_circle(sa,md,pat,r,x,y)
int sa,md,*pat,r,x,y;
{
  int d;

  for(d=0;d<8;d++) {
    g_arc(sa,md,pat,d,r,x,y);
  }
}

/*-----------*/
/*      Draw Arc       */
/*-----------*/
/* sa = screen address */
/* md = mode           */
/* pat = line pattern  */
/* d = direction       */
/* r = circle radius   */
/* x = x coordinate    */
/* y = y coordinate    */
/*-----------*/
void g_arc(sa,md,pat,d,r,x,y)
int sa,md,*pat,d,r,x,y;
{
  static int t[8][2]={
                  -1,0,
                  0,-1,
                  0,1,
                  -1,0,
                  1,0,
                  0,1,
                  0,-1,
                  1,0,
                  };

  if(!r) {
    g_dot(sa,md,x,y);
  }
  else {
    g_poscur(sa,x+t[d][0]*r,y+t[d][1]*r);
    wrcmd(PIGS);
    wrfifo(0x20|d);
    wrparm(((r*46341)+32768)>>16);
```

```
    wrparm(r-1);
    wrparm(2*(r-1));
    wrparm(-1);
    wrparm(0);
    wrcmd(PRAM|0x08);
    wrparm(*pat);
    wrcmd(WDAT|md);
    wrcmd(FIGD);
  }
}

/*-----------*/
/*   Draw Rectangle    */
/*-----------*/
/* sa = screen address */
/* md = mode           */
/* pat = pattern       */
/* x0 = x coordinate   */
/* y0 = y coordinate   */
/* x1 = x coordinate   */
/* y1 = y coordinate   */
/*-----------*/
void g_rct(sa,md,pat,x0,y0,x1,y1)
int sa,md,*pat,x0,y0,x1,y1;
{
  int xt,yt;

  if((x0!=x1)&&(y0!=y1)) {
    if(x0<=x1) xt=x0;
    else xt=x1;
    if(y0<=y1) yt=y0;
    else yt=y1;
    g_poscur(sa,xt,yt);
    wrcmd(FIGS);
    wrfifo(0x40);
    wrparm(3);
    wrparm(abs(y1-y0));
    wrparm(abs(x1-x0));
    wrparm(-1);
    wrparm(abs(y1-y0));
    wrcmd(PRAM|0x08);
    wrparm(*pat);
    wrcmd(WDAT|md);
    wrcmd(FIGD);
  }
  else {
    g_line(sa,md,pat,x0,y0,x1,y1);
  }
}

/*-----------*/
/*  Draw Filled Rectangle  */
/*-----------*/
/* sa = screen address     */
/* md = mode               */
/* pat = pattern           */
/* x0 = x coordinate       */
/* y0 = y coordinate       */
/* x1 = x coordinate       */
/* y1 = y coordinate       */
/*-----------*/
void g_frct(sa,md,pat,x0,y0,x1,y1)
int sa,md,*pat,x0,y0,x1,y1;
{
  int i,xt,yt;

  if((x0!=x1)&&(y0!=y1)) {
    if(x0<=x1) xt=x0;
    else xt=x1;
    if(y0<=y1) yt=y0;
    else yt=y1;
    g_poscur(sa,xt,yt);
    wrcmd(FIGS);
    wrfifo(0x10);
    wrparm(abs(x1-x0));
    wrparm(abs(y1-y0)+1);
    wrparm(abs(y1-y0)+1);
    wrcmd(PRAM|0x08);
    for(i=0;i<4;i++) {
      wrfifo((*pat)>>8);
      wrfifo(*pat++);
    }
    wrcmd(WDAT|md);
    wrcmd(ZOOM);
    wrfifo(0);
    wrcmd(GCHRD);
  }
}

/*-----------*/
/*  Draw Character String  */
/*-----------*/
/* sa = screen address     */
/* md = mode               */
/* sz = size               */
/* sl = slant              */
/* dr = direction          */
/* x = x coordinate        */
/* y = y coordinate        */
/* p = char sting pointer  */
/*-----------*/
void g_str(sa,md,sz,sl,dr,x,y,p)
int sa,md,sz,sl,dr,x,y;
char *p;
{
  int c;
  static int cdt[8][2]={
    0,1,
    1,1,
    1,0,
    1,-1,
    0,-1,
    -1,-1,
    -1,0,
    -1,1,
    };

  while(c=*p++) {
    if(c<0x20|c>0x7f) c='?';
    g_chr(sa,md,sz,sl,dr,x,y,c);
    x+=cdt[dr][0]*(sz+1)*7;
    y+=cdt[dr][1]*(sz+1)*7;
  }
}

/*-----------*/
/*   Draw Character    */
/*-----------*/
/* sa = screen address */
/* md = mode           */
/* sz = size           */
/* sl = slant          */
/* dr = direction      */
/* x = x coordinate    */
/* y = y coordinate    */
/* c = character       */
/*-----------*/
void g_chr(sa,md,sz,sl,dr,x,y,c)
int sa,md,sz,sl,dr,x,y,c;
{
  int i;
  static unsigned char chrgen[128][8]={
    31,17,17,17,17,17,31,0,31,1,1,1,1,1,1,0,
    4,4,4,4,4,4,4,31,0,31,10,4,4,4,10,31,0,
    0,4,13,21,22,4,0,0,31,17,27,21,27,17,31,0,
    16,16,8,8,5,5,2,0,0,0,0,14,17,17,31,0,
    7,3,5,8,8,16,16,0,2,4,8,31,8,4,2,0,
    31,0,0,31,0,0,31,0,4,4,4,4,21,10,4,0,
    4,21,10,4,21,10,4,0,0,0,31,10,4,0,0,0,
    14,17,27,21,27,17,14,0,14,17,17,21,17,17,14,0,
    31,17,17,31,17,17,31,0,14,21,21,29,17,17,14,0,
    14,17,17,29,21,21,14,0,14,17,17,23,21,21,14,0,
    14,21,21,23,17,17,14,0,16,18,12,8,21,5,2,0,
    14,10,10,10,10,10,27,0,16,16,16,31,16,16,16,0,
    31,17,10,4,10,17,31,0,4,4,4,14,4,4,4,0,
    14,17,1,2,4,0,4,0,14,17,17,31,17,17,14,0,
    31,21,21,23,17,17,31,0,0,0,4,10,31,0,0,0,
    16,24,20,18,20,24,16,0,1,3,5,9,5,3,1,0,
    0,0,0,0,0,0,0,0,4,4,4,4,4,0,4,0,
    10,10,0,0,0,0,0,0,10,10,31,10,31,10,10,0,
    4,30,5,14,20,15,4,0,3,19,8,4,2,25,24,0,
    2,5,5,2,21,9,22,0,4,4,4,4,0,0,0,0,0,
```

```
    8,4,2,2,2,4,8,0,2,4,8,8,8,4,2,0,
    4,21,14,4,14,21,4,0,0,4,4,31,4,4,0,0,
    0,0,0,0,4,4,2,0,0,0,0,31,0,0,0,0,
    0,0,0,0,0,0,4,0,0,16,8,4,2,1,0,0,
    14,17,25,21,19,17,14,0,4,6,4,4,4,4,14,0,
    14,17,16,12,2,1,31,0,31,16,8,12,16,17,14,0,
    8,12,10,9,31,8,8,0,31,1,15,16,16,17,14,0,
    28,2,1,15,17,17,14,0,31,16,8,4,2,2,2,0,
    14,17,17,14,17,17,14,0,14,17,17,30,16,8,7,0,
    0,0,4,0,4,0,0,0,0,0,4,0,4,4,2,0,
    16,8,4,2,4,8,16,0,0,0,31,0,31,0,0,0,
    1,2,4,8,4,2,1,0,14,17,8,4,4,0,4,0,
    14,17,21,29,13,1,30,0,4,10,17,17,31,17,17,0,
    15,17,17,15,17,17,15,0,14,17,1,1,1,17,14,0,
    15,17,17,17,17,17,15,0,31,1,1,15,1,1,31,0,
    31,1,1,15,1,1,1,0,30,1,1,1,25,17,30,0,
    17,17,17,31,17,17,17,0,14,4,4,4,4,4,14,0,
    16,16,16,16,16,17,14,0,17,9,5,3,5,9,17,0,
    1,1,1,1,1,1,31,0,17,27,21,21,17,17,17,0,
    17,17,19,21,25,17,17,0,14,17,17,17,17,17,14,0,
    15,17,17,15,1,1,1,0,14,17,17,17,21,9,22,0,
    15,17,17,15,5,9,17,0,14,17,1,14,16,17,14,0,
    31,4,4,4,4,4,4,0,17,17,17,17,17,17,14,0,
    17,17,17,17,17,10,4,0,17,17,17,21,21,27,17,0,
    17,17,10,4,10,17,17,0,17,17,10,4,4,4,4,0,
    31,16,8,4,2,1,31,0,31,3,3,3,3,3,31,0,
    0,1,2,4,8,16,0,0,31,24,24,24,24,24,31,0,
    0,0,4,10,17,0,0,0,0,0,0,0,0,0,31,0,
    2,4,8,0,0,0,0,0,0,0,14,16,30,17,30,0,
    0,2,2,14,18,18,15,0,0,0,30,1,1,1,30,0,
    0,8,8,14,9,9,30,0,0,0,12,18,30,2,12,0,
    0,8,20,4,14,4,4,0,0,0,12,18,18,28,16,12,
    0,2,2,14,18,18,18,0,0,4,0,4,4,4,14,0,
    0,8,0,8,8,8,10,4,0,2,18,10,6,10,18,0,
    0,6,4,4,4,4,14,0,0,0,11,21,21,21,21,0,
    0,0,14,18,18,18,18,0,0,0,14,17,17,17,14,0,
    0,0,14,18,18,14,2,2,0,0,14,9,9,14,8,24,
    0,0,26,6,2,2,2,0,0,0,31,1,14,16,31,0,
    0,4,14,4,4,4,8,0,0,0,18,18,18,18,60,0,
    0,0,17,17,10,10,4,0,0,0,17,17,21,21,10,0,
    0,0,17,10,4,10,17,0,0,0,18,18,18,28,16,12,
    0,0,31,8,4,2,31,0,8,4,4,2,4,4,8,0,
    4,4,4,0,4,4,4,0,2,4,4,8,4,4,2,0,
    0,0,16,14,1,0,0,0,10,21,10,21,10,21,10,0,
};
g_poscur(sa,x,y);
wrcmd(PRAM|0x08);
if(c<0||c>127) c='?';
for(i=0;i<8;i++) {
  wrfifo(chrgen[c][i]);
}
wrcmd(FIGS);
wrfifo(sl<<7|0x10|dr);
wrparm(7);
  wrparm(7);
  wrparm(7);
  wrcmd(WDAT|md);
  wrcmd(ZOOM);
  wrfifo(sz);
  wrcmd(GCHRD);
}

/*------------*/
/*   Position Cursor   */
/*------------*/
/* sa = screen address */
/* x = x coordinate    */
/* y = y coordinate    */
/*------------*/
void g_poscur(sa,x,y)
int sa,x,y;
{
  int t;

  wrcmd(CURS);
  wrparm(sa+=(y*WWDT+x)/16);
  wrfifo((x%16)<<4|sa>>16);
}

/*---------*/
/*  Write Command  */
/*---------*/
/*  c = command    */
/*---------*/
void wrcmd(c)
int c;
{
  while(!(FFE & *necstat)) {}
  *neccmd=c;
}

/*---------*/
/*  Write Parameter  */
/*---------*/
/*  p = parameter    */
/*---------*/
void wrparm(p)
int p;
{
  wrfifo(p);
  wrfifo(p>>8);
}

/*---------*/
/*  Write FIFO  */
/*---------*/
/*  b = byte    */
/*---------*/
void wrfifo(b)
int b;
{
  while(FFF & *necstat) {}
  *necwfifo=b;
}

/*-------*/
/*  Read FIFO  */
/*-------*/
int rdfifo()
{
  while(!(DRY & *necstat)) {}
  return((*necrfifo)&0xff);
}

/*--*/
/* END */
/*--*/

+++
/*========================*/
/*  Graphic Header File  */
/*========================*/

/*-----------*/
/* External Declarations */
/*-----------*/
extern struct spe {int spet[4]};
extern struct spe spt[];
extern int lpt[];

/*-------*/
/* Definitions */
/*-------*/
#define REP          0          /* replace mode */
#define COM          1          /* compliment mode
*/
#define RES          2          /* reset mode */
#define SET          3           /* set mode */
#define OFF      0          /* off value */
#define ON       1          /* on value */
#define TRUE
/* boolean true value */
#define FALSE
/* boolean false value */
#define SOLID    0
* solid line & fill index */
#define VMODE
GA
* video mode desired */
#define DWDT
40
* display width */
```

```
#define DHGT
80          /* display higth */
#define MHGT    41          /* menu window
height */
#define WWDT    1200
* window width */
#define USHGT   800
* upper screen hight */
#define LSHGT   70          /* lower screen
height */
#define UPPER
/* upper screen address */
#define LOWER   UPPER+USHGT*WWDT/16
* lower screen address */

/*-----------*/
/* Control Parameters */
/*-----------*/
#define NEC
x800000
* NEC control address */
#define GRSWP
EC
* read status, write parameter address */
#define GRFWC   NEC+2          /* read fifo,
write command address */

/* VGA init string for 640/480 display with
24mhz clock */
#define VGA 40,3,3,2,4,7,480,20

/* EGA2 init string for 704/350 display with
18.432mhz clock */
#define EGA2 44,3,3,4,2,1,350,8

/* EGA1 init string for 640/350 display with
18.432mhz clock */
#define EGA1 40,3,3,6,4,1,350,8

/* CGA init string for 640/240 display with
12mhz clock */
#define CGA 40,3,3,4,2,7,240,10

+++

*====================================*
* Makefile for Graphic Demo Program *
*====================================*

PRG = /h0/cmds/gdemo
RDIR = /dd/rels
CFLAGS = -t=/dd -w=/dd/lib ^64
FILES = ${RDIR}/gdemo.r ${RDIR}/gdglb.r ${RDIR}/
gdrv.r
${PRG} : ${FILES}
     cc ${FILES} ${CFLAGS} -f=${PRG}
```

Bit-Bucket

By: All of us

"Contribute Nothing · Expect Nothing", DMW '86

Continued from last month page 49 Oct.1988 issue.

DO

A FLEX-09 Batch File Processor

By: Dave Howland

Here, DO will jump to the label 'reporterror' if any Flex command in the body of the batch file returns an error.

PAUSE <message>

The text following the PAUSE command is a message to the user prompting some action, eg to change a disk. The message will be output to the terminal, as described for the NOTE command, followed by the message "Press any key to continue ...". DO then waits for the user to enter a character before continuing.

The following example prompts the user for a new disk :

PAUSE Insert system disk into drive 0

REM <comment> ! <comment>

Either of the above are comments in the batch file and are not processed by DO, except to replace command line parameters.

SHIFT

This command causes the command line parameters 1 to 9 to be shifted by one, i.e. %2 becomes %1, %3 becomes %2 etc. The following example shows the effect of SHIFT :

Parameters before SHIFT :

%0 = test %1 = one %2 = two %3 = three %4 to %9 = null string

Parameters after SHIFT :

%0 = test %1 = two %2 = three %3 to %9 = null string

The use of this can be demonstrated in the following batch file which deletes a list of files given as command line parameters :

```
ECHO OFF ! ! Delete files on the command line ! IF '%1 = ' GOTO help @loop  IF EXIST %1 THEN
delete %1   Y%c  ELSE   NOTE %1 doesn't exist  ENDIF  SHIFT IF NOT '%1 = ' GOTO loop GOTO
exit ! @help NOTE Usage : DO DEL <file> <file> ... @exit
```

Note the use of extra quote characters on the first and third IF lines to allow testing for a null parameter. The first IF checks for no specified files, in which case the help information is output; the third IF checks for %1 becoming null, indicating that all of the files have been processed.

4.4 Chaining DO Batch Files

It is possible to chain one batch file from another by calling DO in the first batch file with the required filename and parameters for the second batch file. Once the second batch file finishes, DO will return to Flex; it does not return to the first file.

For example, a batch file may contain the following lines :

! ! format file, print it and return to Flex ! do format %1.doc

where %1 is a filename parameter passed into the batch file from the command line.

5. MEMORY REQUIREMENTS

DO consists of two parts; a non-resident part which loads into the transient program area (at $C100) and a resident part which loads at $0100 but is moved to the top of memory by the non-resident part. The non-resident part is used to interpret the command line parameters and to relocate the resident part, adjusting MEMEND appropriately; it may be overwritten once DO is installed. The resident part interprets the batch file and executes the commands contained in it.

The resident part of DO occupies about 2300 bytes, so this much less memory will be available to user programs. However, so long as the user programs take account of MEMEND this should not be a problem.

On exit, DO will restore MEMEND to its previous value UNLESS some other program loaded by DO has modified MEMEND, in which case DO cannot be removed from memory and a warning message is output to the terminal.

6. CAUTIONS

There are a number of important points to bear in mind when running batch files :

- It is not advisable to load other memory resident programs which also modify MEMEND unless they remove themselves before DO exits, otherwise DO cannot remove itself from memory. If this situation occurs, a warning message will be output to the terminal and DO will exit to Flex without restoring MEMEND.

- Because DO changes the INCH vector (at $CD0A), the I utility should not be used from inside a batch file; the O utility may be used however.

- Because DO saves the TTYSET PS flag, any attempt to change this inside a batch file may be over-ridden when DO exits and restores the original value.

7. EXAMPLE BATCH FILE

This batch file example can be used to provide some measure of automation to the assembler program development cycle. It is called with one parameter, which is either '+h' to display help information, or a filename (with no extension) which is the source file to be developed. The batch file automatically assigns the source file extension '.txt'. Note that it is advisable to specify a drive number to ensure that all files are put onto the same disk.

```
ECHO OFF ! ! DEV.BAT ! Batch file for development of assembler programs ! Written by Dave
Howland ! Version 1.1, 21st December 1987 ! ! If no parameters or '+h', output help info ! IF '%1 = '
GOTO help IF %1 = +h GOTO help ! ! Perform loop until user does not wish to ! continue the develop-
ment cycle ! @loop   edit %1.txt   %t   NOTE Assemble source file   IF %c = y THEN     IF EXIST
%1.cmd THEN      delete %1.cmd      yy    ENDIF    asmb %1.txt %1.cmd +ls   ENDIF   NOTE
Continue development cycle IF %c = y GOTO loop GOTO exit ! ! Output help info ! @help NOTE Usage
: do dev <file> ! ! Single exit point ! @exit
```

EOF

To be continued next month!

MOTOROLA INC.

CONTACTS
Shernaz Daver
Cunningham Communication, Inc.
(408) 982-0400

Dean Mosley
Microprocessor Products Group
(512) 440-2839

APPLE INTRODUCES MACINTOSH BASED ON MOTOROLA'S 68030

68030 Increases Performance and Maintains Compatibility in Macintosh Line

AUSTIN, Texas, Sept. 28, 1988 — Apple Computer, Inc. announced that its new Macintosh computer incorporates two Motorola microprocessors, the 68030 (030) and the 68882 (882). The new machine, called the Macintosh IIx, uses Motorola's 16 MHz 030 as the central processing engine and the 16 MHz 882 as a floating-point math coprocessor.

The Macintosh IIx is the latest computer in the Macintosh line to use Motorola's 68000 family of microprocessors. Motorola's 68000 processor drives the Macintosh Plus and SE. The 68020 processor and 68881 math coprocessor power the Macintosh II, a widely accepted computer for business and engineering applications. Compatibility between all processors in the 68000 family allows for easy migration of software from early Macintosh computers to the new 030-based Macintosh IIx.

"Apple continues to innovate with our 68000 family of microprocessors," said Murray A. Goldman, senior vice president and general manager of Motorola's Microprocessor Products Group (Austin, Texas). "Our evolving 68000 family will provide Apple with the price and performance needed to push the limits of the personal computer market."

Motorola's 030 provides a fully integrated solution with a dual bus architecture, paged memory management unit (PMMU) and separate 256-byte data and instruction caches on a single chip. The PMMU is required in multiuser and multitasking environments and supports sophisticated operating systems such as A/UX. The data and instruction caches increase system performance by enabling quick access to the most recently used data and instructions.

The 882 floating-point math coprocessor provides sophisticated numeric functions on a single chip. Floating-point coprocessors are used to speed mathematical calculations in a variety of business, financial and engineering applications. The 882 is software and pin compatible with its predecessor, the 68881 (881). The 882, the first chip to break the 2 million Whetstone barrier, gives users twice the performance of the 881.

GENERAL MICRO SYSTEMS
INCORPORATED

For further information:
Scott Bowman (714)625-5475

FOR IMMEDIATE RELEASE

PHOTOS ENCLOSED
Color & B&W

33 MHz 68030 32-BIT CPU BOARD WITH 1 MB NO-WAIT SRAM FOR VME MULTIPROCESSING

Montclair, Calif., September 21, 1988-- A VMEbus CPU board based on the 33 MHz, 32-bit 68030 with dual cache memory and MMU and with a 68882 co-processor, up to 1 MByte of dual-ported zero-wait-state SRAM, and unique interrupt and booting capabilities to support high speed multiprocessing systems, is now available from General Micro Systems Inc.

The GMSV17 is based on the 68030 CPU device. This provides up to 33 MHz speed (20 and 25 MHz available) with MMU; its dual cache memory provides an instruction cache and a data cache with Harvard architecture. A 68882 floating point co-processor further supports full 32-bit processing.

Up to 1 MByte of zero-wait-state SRAM is on-board. This local memory can be protected from unwanted writes yet add-on DRAM or SRAM can still provide dual porting to the rest of a multiprocessor system. Up to 512 KByte of EPROM (128 KBytes of EEPROM) is also available, for program codes. Mezzanine modules can supply an additional 1 MByte of dual-ported SRAM or 4/16 MByte of dual-ported DRAM on a local bus with the final module still requiring only a single VME slot.

The board both originates and services interrupts on the VMEbus. Unique location monitor/mailbox interrupt techniques allow real-time multiprocessing. A 68155 Bus Manager provides interrupts and manages mailbox locations, so arbitration between software and hardware interrupts is automatic and failsafe.

A unique Bus Master Boot requires only one boot PROM for an entire multiple CPU system. This lets code execute faster, provides smoother system integration, and makes PROM sockets available for extra RAM.

The GMSV17's configuration is fully programmable. It can be dynamically configured or can be pre-configured to aid production situations. The configuration controller provides 3 timers which can also function as a real time clock.

Two multiprotocol serial ports are also provided which can meet RS232/RS422/485 specifications.

The board also accepts SAM™ (Special Application Module) mezzanine modules, adding a variety of functions on local busses, eliminating the bottleneck of many VME systems, too much traffic on the VMEbus. SAM modules include various I/O (serial, parallel), control functions (SCSI, floppies, Ethernet, IEEE 488) and even a second VSB bus which supports both master and slave access to every CPU and memory in the system. Addition of SAM mezzanine modules does not require an additional slot.

The GMSV17 is a double Eurocard 9.2" x 6.3" (234 mm x 160 mm) requiring a single slot in a VME card cage. The board also is available for extended temperature applications.

Available operating systems for the GMSV17 include PDOS, PSOS, VRTX, OS-9 and VMEPROM. Additional software includes a full featured de-bugger and on-board diagnostics.

PRESS RELEASE

GESPAC Inc.
50 West Hoover Ave.
Mesa, Arizona 85210
Tel. (602) 962-5559
Fax. (602) 962-5750

Reader Contact: Don Bizios
Editorial Contact: Cosma Pabouctsidis

FOR IMMEDIATE RELEASE

FULL CMOS, EXTENDED TEMPERATURE
68000 BOARD CONSUMES ONLY 70 mA

Mesa, AZ, September 15, 1988-- GESPAC has released an all CMOS single board computer for the G-64 bus, designed around the 68HC000 microprocessor, featuring very low power consumption and high powered 16-bit processing capability. The MPL-4079 is a 6" by 6" Euroboard incorporating many peripheral devices, including 2 serial ports, a 4-channel A/D converter, parallel I/O, a clock, EPROM and battery-backed RAM. Many other functions can be added to this computer, as it is fully compatible with the wide range of I/O cards available for the G-64 bus. Software development is fully supported for the OS-9 real-time operating-systems.

The introduction of the MPL-4079 computer increases the attractiveness of the G-64 bus. Already known for its compact size, low cost and high performance, the G-64 bus will now be suitable for low power consumption, single-supply applications, with the added option of extended industrial temperature range. The board requires a single 5 V supply, drawing a mere 70 mA.

The MPL-4079 uses a 68HC000 CPU running at 8 MHz, with higher speed grades available. Up to 768 kB of EPROM and SRAM can be installed in the six available sockets, with a full battery back-up of RAM, thus permitting non-volatile storage.

The CPU and memory available on board are sufficient to run most applications running under the OS-9 operating system, with drivers available for all the peripherals, and full facilities for program down-loading through the serial port, and symbolic debugging. The user could also develop programs on an IBM PC and download the code to the board.

Industrial real-time applications are fully supported by the presence of a four-channel, 8-bit A/D converter, featuring a 40 microsecond conversion time. There are also 20 parallel I/O lines available for logic level inputs. Two RS-232 serial ports are provided for communications; the baud rates are programmable. The MPL-4079 is equipped with such useful local peripherals as power-failure detection, watchdog timer, two 16-bit timers and a battery-backed real-time clock with calendar.

WINDRUSH
Micro Systems Ltd.

Windrush Micro Systems (Rest Office)
North Walsham, Norfolk NR28 9SA
Tel: (0692) 404086
Telex: 975548 WMICRO G
MAPCON approved consultants

PRESS RELEASE

PROCON-IV ... Bridges the Gap between PLCs and Micros

Windrush Micro Systems Limited are pleased to announce the immediate availability of their PROCON-IV family of process controllers and monitors.

The PROCON-IV PDLC is a 68000 based Programmable Data Logger Controller designed to bridge the gap between PLCs and Microprocessor based systems.

Many applications, particularly those involving real-time analogue signals, often exceed the abilities of PLCs and are not sufficient justification to install a full-blown microprocessor based system.

Windrush can provide PDLCs fully programmed as a plug-in 'Black Box' for customers who do not wish to become involved with programming the system themselves. These solutions can communicate with a PLC via the digital I/O signals or via the RS-232 port as required.

PROCON-IV can be configured with a choice of input and output modules. Floppy and hard disk options are also available for data logging and process monitoring applications. 32-bit 68020 processor modules with 68881 math-coprocessor support are also available for heavy 'number crunching' front end processing.

Programming facilities can be integrated into the PDLC or the programming can be done on a seperate 'Development System'. The user has a choice of BASIC, PASCAL, C, FORTRAN and PL/S-68K as system programming languages. If the application requires a real-time multi-tasking executive OS-9/68K Industrial can be supplied.

A 'standard' PROCON-IV configuration which includes an 8MHz MC68000, 256Kb Non-volatile ram, 128Kb ROM, clock calender, RS-232C serial port, 48 optically isolated inputs, 48 optically isolated outputs, 32 analogue inputs sells for less than £1995 (one off).

For further information contact Bill Dickinson at (0692) 404086

Commodore NEWS

1200 Wilson Drive
West Chester, PA 19380
(215) 431-9100

For further information
contact Lori Cross at
Fleishman-Hillard
(213) 629-4974

COMMODORE INTRODUCES 8 MEGABYTE RAM EXPANSION CARD FOR THE AMIGA 2000

West Chester, PA, September 19, 1988 -- Commodore Business Machines, Inc. today introduced the A2058 memory expansion card for the Commodore Amiga 2000. When fully configured, the new 8 megabyte RAM expansion card provides the maximum possible memory for the Amiga 2000 while using a single expansion slot. The standard A2058 card comes configured with 2 megabytes of RAM and users can add an additional 6 megabytes of RAM using 1 megabit DRAM chips.

"The A2058 gives users additional RAM required for more sophisticated Amiga 2000 applications such as 3-D animation and graphic design," said David Archambault, Commodore director of project marketing. "In addition, the increased RAM enhances the multi-tasking features of the Amiga by permitting additional tasks to run simultaneously."

The Amiga 2000 system auto-configures for the additional memory which can be accessed and utilized by the CPU.

The A2058 memory expansion card will be available through authorized Commodore Amiga dealers and has a suggested retail price of $799.

The Commodore Amiga 2000 is a powerful, multi-tasking, graphics-oriented computer which excels in professional graphics and video applications. The unit features a built-in 3 1/2-inch disk drive, custom sound, animation and graphics chips, a detachable 92-key keyboard with separate numeric keypad and ten function keys and a two-button optomechanical mouse.

The open architecture design of the Amiga 2000 allows extensive internal expansion with multi-processor, multi-DOS options and includes seven full-size expansion slots configured as Amiga or standard PC XT/AT slots, a CPU expansion slot for 68020/68030 and/or math coprocessors and video expansion slot.

The Amiga 2000 is part of Commodore's complete line of graphics-oriented personal computers. In addition, Commodore also manufactures the IBM-compatible Professional Series III and Commodore Colt computers. The Commodore 64 and Commodore 128 have an installed user base over ten million worldwide.

# # #

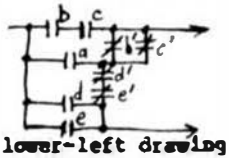
RBASIC FOR 68000 SK*DOS

Announcing the availability of RBASIC, version 1.5, our full-featured BASIC interpreter for SK*DOS running on 68000 machines. Upwards compatible with 6809 XBASIC (by Technical Systems Consultants) and our own 6809 RBASIC, though some modification will be necessary for some instructions, such as I/O redirection on Channel #0 and USR.
Some of the extra features include choice of Degrees or Radians mode, plus an ARC statement, for trig functions, APPEND and BLOCK-delete commands, AND A BUILT-IN LINE-EDITOR for intelligent terminals with Cursor-Left and Cursor-Right controls.
Price in Canada is $125 Canadian plus $5 shipping and handling, or US $99.95 plus $5 within the continental US. Elsewhere, the price is US $99.95 plus $10 S/H. Payment may be made directly to Micronics Research by International Money Order payable to R. Jones (sorry, no credit cards).
Of course, the 6809 version (currently 2.5) is still available directly from us at the terms set out above.
Please Specify Disk-Format required, unless specified otherwise RBASIC will be shipped on a 5" Floppy, 80-Track, Single-Sided, Single-Density.

Dear Don,

It's embarrassing, but here's a summary of my typos in the Sept 88 issue of 'Logically Speaking', so readers can update their lessons.

1. P. 22 last sentence "n-input" NOT "n-iput"
2. P. 23 final expression. Change to
(a+b')(bc+a'c') = abc+a.a'c+b'.bc+a'b'c' = abc+a'b'c'
3. P.24 centre drawing of 4d should look like



4. P. 24 lower-left drawing - blank contact should have a label "a".
5. P. 26 para 1 sentence 2 "... term ab'c' ..." NOT ab'c.
6. P. 26 penult para Insert "in the 1-decoding" after the words "... a phi is read as '1' ..."
7. P. 27 1st expression = a'b + ac(b' + d') NOT (b' + c')
8. P. 27 roughly centre page would be better with parens, thus a'b OR (ac but NOT abcd).
9. P. 27 immediately above K-map, following the words "... expression of 65a." add the following :
The reason we can delete "ac" from inside the parens is, of course, because
ac(acxy)'=ac(a'+c'+x'+y')=aca'+acc'+acx'+acy'
where the first two terms of the final expression are obviously self-cancelling, reducing to the same form as if we'd begun with ac(xy)'
10. p. 36 near top, amend to "c.(bcd')' reducing to c.(bd')' and finally ..."
11 p. 36 final K-map needs "cd" added to heading.

Sorry about all that! These things seem to have a way of creeping in despite my best efforts!!

Since my last letter on the PTR function, I've been sharply reminded that PTR is used a lot, particular with the USR function. Keeping in mind that only an Integer can normally be passed as a parameter in USR, some readers have adopted the useful technique of getting the address of a Floating-Point or String variable via PTR, and passing this address to a machine-language sub-routine as a pointer to a FP or String. Sneaky, huh? As a result of this, I've re-instated PTR into RBASIC, and added a new function RNME whereby a Global Name Change can be made, even to subscripted variables.

Don Williams,
68 Micro Journal,
5900 Cassandra Smith Road,
Hixson, TN 37343

Sincerely,

R. Jones
President

Classifieds

As Submitted - No Guarantees

MUSTANG-020 20Mhz with 68881. OS9 Professional Package & C $2900. Call Tom (615)842-4600.

AT&T 7300 UNIX PC, UNIX V OS, 1MB Memory, 20 MB Hard Disk, 5" Drive, Internal Modem, Mouse. Best Offer Gets It.

S+System with Cabinet, 20 Meg Hard Disk & 8" Disk Drive with DMAF3 Controller Board. 1-X12 Terminal $3400.

HARD DISK 10 Megabyte Drive - Seagate Model #412 $275.
3-Dual 8" drive enclosure with power supply. New in box. $125 each.
5-Siemens 8" Disk Drive, $100 each.

Tano Outpost II, 56K, 2 5" DSDD Drives, FLEX. MUMPS, $495.
QUME QVT-102 terminal, like new, amber screen $250. or best offer.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09CPU Card. $900 complete.
Tom (615) 842-4600 M-F 9AM to 5PM EST

***

S+ System with cabinet, 20 Meg Hard Disk, 8" Floopy, Tape Backup, 512K, 16 Serial Ports. $2000 or best ofer. 5 X12 Treminals $500 ea. or best offer
Terry (716) 235-4631

***

Special Liquidation

30 Motorola MVME 133, CPU Module, 68020, 12.5 Mhz. 1 Meg DRAM, $675 ea.
30 Motorola MVME 225-1, 1 Meg Memory Module, $380 ea.
30 Motorola MVME 320A, Winchester/Floopy Controller, ST506 compatible, $1295 ea.
30 Motorola MVME 332, Intelligent 8 Channel Serial Communications Module, 68010 supports multiple protocols, $1795 ea.
35 Electronic Solutions 7 Slot VME Desktop Enclosures w/ 325 watt power supply, supports up to 4 mass storage devices, $2000 ea.
2 Configured Systems, 40 Meg Seagate Winchester, 1 Meg Floopy, $3200 ea.
Call or write J.G. (602) 951-3373, RPG, POBox C6000, Suite 162, Scottsdale, Az. 85261 or
Tom Williams, (615) 842-4600.

ATARI & AMIGA CALL

As most of you know, we are very sensitive to your wishes, as concerns the contents of these pages. One of the things that many of you have repeatedly written or called about is coverage for the **Atari & Amiga™** series of 68000 computers.

Actually we haven't been too keen on those systems due to a lack of serious software. They were mainly expensive "game-toy" systems. However, recently we are seeing more and more honest-to-goodness serious software for the Atari & Amiga machines. That makes a difference. I feel that we are ready to start some serious looking into a section for the Atari & Amiga computers. Especially so since OS-9 is now running on the Atari (review copy on the way for evaluation and report to you) and rumored for the Amiga. Many of you are doing all kinds of interesting things on these systems. By sharing we all benefit.

This I must stress - *Input from you on the Atari & Amiga*. As most of you are aware, we are a "contributor supported" magazine. That means that YOU have to do your part. Which is the way it has been for over 10 years. We need articles, technical, reviews of hardware and software, programming (all languages) and the many other facets of support that we have pursued for these many years. Also I will need several to volunteer to do regular columns on the Atari & Amiga systems. Without constant input we can't make it fly! So, if you do your part, we certainly will do ours. How about it, drop me a line or give me a phone call and I will get additional information right back to you. We need your input and support if this is to succeed!

DMW

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibbles
OS9 USER NOTES

Information for the BEGINNER to the PRO, Regular or CoCo OS9

Using OS9
HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS, OS9 STANDARDS, Generating a New Bootstrap, Building a new System Disk, OS9 Users Group, etc.

Program interfacing to OS9
DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION, "SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages
Assembly Language Programs and Interfacing; Basic09, C, Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include
No typing all the Source Listings in. Source Code and, where applicable, assembled or compiled **Operating Programs.** The Source and the Discussions in the Columns can be used "as is", or as a "Starting Point" for developing your **OWN** more powerful Programs. Programs sometimes use multiple Languages such as a short Assembly Language Routine for reading a Directory, which is then "piped" to a Basic09 Routine for output formatting, etc.

BOOK $9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)
Deluxe Binder - - - - - - - - - - $5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - $14.95
2-5" SS, DD Disks - - - $24.95

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERM EM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC

NOTE: .C1,.C2, etc.=Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included is ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

Shipping & Handling $3.50 per Book, $2.50 per Disk set
Foreign Orders Add $4.50 Surface Mail
or $7.00 Air Mail
If paying by check - Please allow 4-6 weeks delivery
*** All Currency in U.S. Dollars**

Continually Updated In 68 Micro Journal Monthly

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343

(615) 842-4601
Telex 5106006630

"FLEX is a trademark of Technical Systems Consultants
"OS9 is a trademark of Microware and Motorola
"68' Micro Journal is a trademark of Computer Publishing Inc.

!!! Subscribe Now !!!
68 MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard ☐ VISA ☐

Card #_____ Exp. Date_____

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: $_____

Name _____

Street _____

City _____ State ____ Zip _____

Country _____

My Computer Is: _____

Subscription Rates

U.S.A.: 1 Year $24.50, 2 Years $42.50, 3 Years $64.50

***Foreign Surface: Add $12.00 per Year to USA Price.**

***Foreign Airmail: Add $48.00 per Year to USA Price.**

***Canada & Mexico: Add $9.50 per Year to USA Price.**

***U.S. Currency Cash or Check Drawn on a USA Bank !**

68 Micro Journal
5900 Cassa dra Smith Rd.
POB 849
Hixson, TN 37343

Telephone 615 842-4600
Telex 510 600-6630

Reader Service Disks

Disk- 1	Filesort, Minicat, Minicopy,Minisms, **Lifetime, **Poetry, **Foodlist, **Diet.
Disk- 2	Diskedit w/ inst.& fixes, Prime, *Prmod, **Snoopy, **Football, **Hexpawn,**Lifetime.
Disk- 3	Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.
Disk- 4	Mailing Program, *Finddat, *Change, *Testdisk.
Disk- 5	*DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
Disk- 6	**Purchase Order, Index (Disk file indx).
Disk- 7	Linking Loader, Rload, Harkness.
Disk- 8	Cnest, Lanpher (May 82).
Disk- 9	Datecopy, Diskfix9 (Aug 82).
Disk-10	Home Accounting (July 82).
Disk-11	Dissembler (June 84).
Disk-12	Modem68 (May 84).
Disk-13	*Initmf68, Testmf68,*Cleanup,*Dskalign,Help,Date.Txt.
Disk-14	*Init, *Test, *Terminal, *Find, *Diskedit, Init.Lib.
Disk-15	Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
Disk-16	Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
Disk-17	Match Utility, RATBAS, A Basic Preprocessor.
Disk-18	Parse.Mod, Size.Cmd (Sept. 85 Armstrong),CMDCODE, CMD.Txt (Sept. 85 Spray).
Disk-19	Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
Disk-20	UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist). Dragon.C, Grep.C, LS.C, FDUMP.C.
Disk-21	Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
Disk-22	Read CPM & Non-FLEX Disks. Fraser May 1984.
Disk-23	ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven. Language Recognition Utility, Anderson March1986.
Disk-24	68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
Disk-25	KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
Disk-26	Compacta UniBoard review, code & diagram, Burlison March '86.
Disk-27	ROTABIT.TXT, SUMSTEST.TXT, CONDATA.TXT, BADMEN.TXT.
Disk-28	CT-82 Emulator, bit mapped.
Disk-29	**Star Trek
Disk-30	Simple Winchester, Dec.'86 Green.
Disk-31	*** Read/Write MS/PC-DOS (SK*DOS)
Disk-32	Heir-UNIX Type upgrade - 68MJ 2/87
Disk-33	Build the GT-4 Terminal - 68MJ 11/87 Condon.
Disk-34	FLEX 6809 Diagnostics, Disk Drive Test, ROM Test, RAM Test - 68MJ 4/88Korpi.

NOTE:

This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC
*** Denotes 68000 - 6809 no indicator.

8" disk $19.50
5" disk $16.95

Shipping & Handling -U.S.A. Add: - $3.50
Overseas add: $4.50 Surface - $7.00 Airmail

68 MICRO JOURNAL
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4600 - Telex 510 600-6630

K-BASIC™

The Only 6809 BASIC to Binary Compiler for OS-9 FLEX or SK*DOS
Even runs on the 68XXX SK*DOS Systems*

Hundreds Sold at Suggested Retail:

~~$199.00~~

• 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

• 6809 - FLEX users can compiler their BASIC source files to a regular FLEX ".CMD" file. Much faster execution.

• 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

Don't get caught up in the *"Learn a New Language"* syndrome - Write Your Program in BASIC, Debug It in BASIC and Then Compile it to a .CMD Binary File.

For a LIMITED time save over 65%... This sale will not be repeated after it's over! *

SALE SPECIAL:

$69.95

SPECIAL

Thank-You-Sale

Only From:

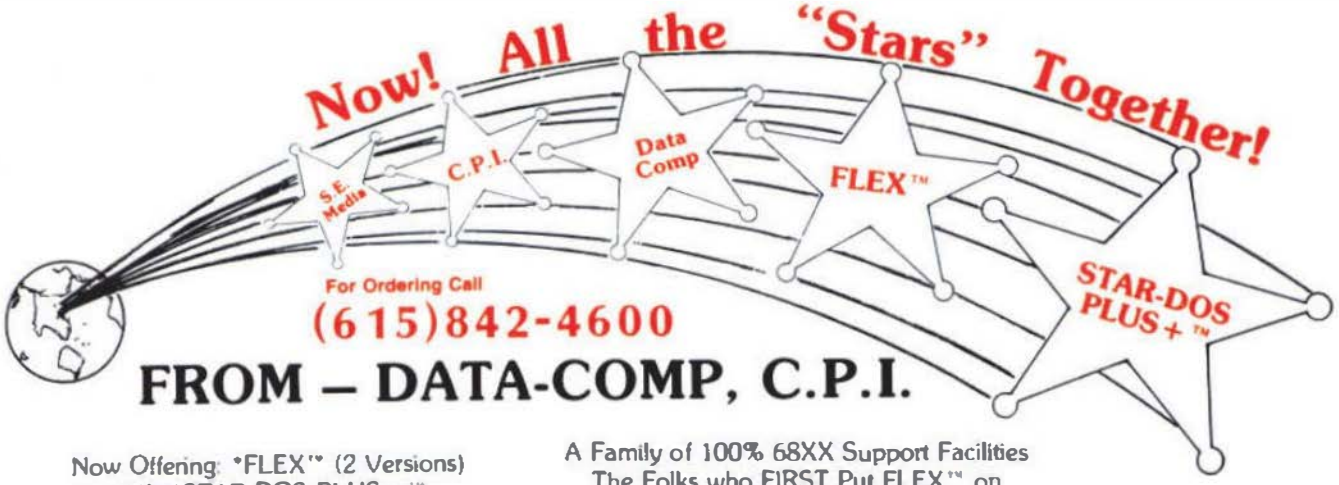
CPI **S.E. Media™**

5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6530

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000

* K-BASIC will run under 68XXX SK*DOS in emulation mode for the 6809.

Price subject to change without notice.



FROM – DATA-COMP, C.P.I.

Now Offering: *FLEX™ (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler
Complete with Manuals
Reg. $250.00 **Only $79.95**

STAR-DOS PLUS+
- Functions Same as FLEX
- Reads - writes FLEX Disks
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

$34.95

FLEX-CoCo Jr.
without TSC Editor & Assembler
$49.95

PLUS

ALL VERSIONS OF FLEX & STAR-DOS• INCLUDE

+ Read-Write-Dir RS Disk
+ Run RS Basic from Both
+ More Free Utilities
+ External Terminal Program
+ Test Disk Program
+ Disk Examine & Repair Program
+ Memory Examine Program
+ Many Many More!!!

TSC Editor
Reg $50.00
NOW $35.00

TSC Assembler
Reg $50.00
NOW $35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES SYSTEM WITH POWER SUPPLY,CABINET,DISK DRIVE CABLE,J&M NEW DISK CONTROLLER JFD-CP WITH J-DOS,RS-DOS OPERATING SYSTEMS.	$469.95
* Specify What CONTROLLER You Want J&M, or RADIO SHACK	
THINLINE DOUBLE SIDED DOUBLE DENSITY 40 TRACKS	$129.95

Verbatim Diskettes

Single Sided Double Density	$ 24.00
Double Sided Double Density	$ 24.00

Controllers

J&M JFD-CP WITH J-DOS	$139.95
WITH J-DOS, RS-DOS	$159.95
RADIO SHACK 1.1	$134.95
RADIO SHACK Disk CONTROLLER 1.1	$134.95

Disk Drive Cables

Cable for One Drive	$ 19.95
Cable for Two Drives	$ 24.95

MISC

64K UPGRADE FOR C,D,E,P, AND COCO 11	$ 29.95
RADIO SHACK BASIC 1.2	$ 24.95
RADIO SHACK DISK BASIC 1.1	$ 24.95
DISK DRIVE CABINET FOR A SINGLE DRIVE	$ 49.95
DISK DRIVE CABINET FOR TWO THINLINE DRIVES	$ 69.95

PRINTERS

EPSON LX-80	$289.95
EPSON MX-70	$325.95
EPSON MX-100	$495.95

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD	$ 89.95
8149 32K EXPAND TO 128K	$169.95
EPSON MX-RX-80 RIBBONS	$ 7.95
EPSON LX-80 RIBBONS	$ 5.95
TRACTOR UNITS FOR LX-80	$ 39.95
CABLES & OTHER INTERFACES CALL FOR PRICING	

DATA-COMP
5900 Cassandra Smith Rd.
Hixson, TN 37343

MasterCard VISA

SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. $2.50

(615)842-4600
For Ordering
Telex 5106006630

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the $500.00 OS-9 C compiler! This offer won't last forever!

NOT 128K, *NOT* 512K

FULL 768K No Wait RAM

The MUSTANG-08™ system took every hand from all other 68008 systems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-68K™ and/or Peter Stark's SK*DOS™. SK*DOS is a single user, single tasking system that takes up where *FLEX™ left off. SK*DOS is actualy a 68XXX FLEX type system *(Not a TSC product.)*

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a speed *whiz on disk I/O.* Fact is the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks!

SALE!!

System includes OS-9 68K or SK*DOS - Your Choice

Specifications:

CPU	MC68008	12 Mhz
RAM	768K No Wait States	256K Chips
PORTS	4 - RS232 2 - 8 bit Parallel	MC68681 DUART MC6821 PIA
CLOCK	MK48T02	Real Time Clock Bat. B/U
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Now more serial ports - faster CPU Battery B/U - and $850.00 OS-9 Professional with C compiler included!

***$400.00** See Mustang-02 Ad - page 5 for trade-in details

Now even faster!
with 12 Mhz CPU

C Compile times: OS-9 68K Hard Disk	
MUSTANG-08 8 Mhz CPU	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec

25 Megabyte Hard Disk System

$2,398.90

Complete with PROFESSIONAL OS-9 includes the $500.00 C compiler, PC style cabinet, heavy duty power supply, 5" DDDS 80 track floppy, 25 MegByte Hard Disk - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 MegaBytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC MUSTANG-08 is a trademark of CPI

MUSTANG-08

LOOK

					Seconds	32 bit Integer	Register Long
Other	68008	8 Mhz	OS-9	68K	18.0	9.0	
MUSTANG-08	10 Mhz		OS-9	68K	9.8	6.3	

```
Main()
{
                    /* int i; */
C Benchmark Loop    register long i;
                    for (i=0; i < 999999; ++i);
}
```

Data-Comp Division

A Decade of Quality Service"

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road

Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

* Those with SWTPC Hi-density FLEX 5" - Call for special info.